(2)

放送用実時間映像信号処理システムに関する研究

八木　伸行

目次

[1] 序論

1．1　研究の背景と意義

１９５０年代の文字パターン認識の研究に始まったディジタル画像処理技術は，半導体技術，コンピュータ技術の進展とともに，社会のニーズの合わせて発展し，現在では医学，リモートセンシング，ＦＡ（Factory Automation），ＯＡ（Office Automation）など幅広い分野で使われている．

このようにますます重要となる画像処理であるが，処理内容の高度化，実時間処理が求められてきている．さらに最近は，動画像も扱うようになり，ビデオレートで処理することも求められつつある．このような要求に対して，汎用のコンピュータでは対応しきれないため，専用の画像処理装置が研究開発されている[1]．

一方，放送における映像信号処理は，従来アナログで処理されてきたが，１９７２年のディジタルＴＶ方式変換装置を手始めに，ディジタル方式への移行が始まった[2]．当初は，アナログ回路では実現できない機能をディジタルで実現するといった方法が採られていたが，高画質，高安定などのメリットから，あらゆるスタジオ用映像機器のディジタル化が進められるようになった．ところが，これらは内部処理のディジタル化であって，入出力は相変わらずアナログ映像信号のままであった．このため，機器を接続して処理を重ねると，Ａ／Ｄ，Ｄ／Ａが何回も繰り返されることになり，高画質化などのディジタル化のメリットが生かされないままであった．そこで，トータルディジタル化，すなわち入出力も含めたシステム全体のディジタル化が，時代の要請となった．このような状況の中，１９８２年にＣＣＩＲ（国際無線通信諮問委員会）において，ディジタル規格が決められ，スタジオ設備のトータルディジタル化が図られるようになった．しかし，アナログシステムを単純にディジタルシステムに置き換えただけでは，高画質化，高安定化だけの改善に留まり，投資に対して得られるものがあまりに少ない．ディジタル化によるメリットは，アナログ方式では得られなかった柔軟性に富んだシステム構成が可能になり，システム構成法の変革をもたらすことが一番大きいのではないだろうか．

従来，放送局のスタジオにおける映像信号処理は，ビデオスイッチャや特殊効果装置など専用機器を組み合わせて行っており，新しい映像効果や機能に対しては，要求ごとに新規に専用機器を開発してきた．しかし，このような方式では，放送メディアが増大し，番組制作手法が多様化し，処理内容が複雑多岐にわたるようになっている昨今，経済性，即応性の面でとても追いつかない状況になってきている．例えば，

（１）新しい映像表現のために次々と特殊効果が開発されてきたが，飽きられて使われなくなってしまうものも多く，コスト的にも問題がある．
（２）画質に難点があるものの再収録の困難なビデオテープを後処理で救いたい場合があるが，障害の状況はテープ毎に異なり，多様な障害に対応するために機器をいちいち開発することは困難であり，即応性に欠ける．
などである．

これらに対して，同一のハードウェア上で，ソフトウェアを入れ換えることによって対応するようにすれば，広範な放送局のニーズに，柔軟にしかもすばやく対応できるようになるはずである．ハードウェア（ワイヤードロジック）のソフトウェア化は，時代の流れに沿うものであり，機器の制御がワイヤードロジックによる制御からマイクロコンピュータによるプログラム制御に変わって，一つのコントローラでプログラムを変えることにより，きめ細かにかつ柔軟に対応できるようになったのと同じである．

プログラムで処理する装置といえば，従来よりコンピュータがある．また，用途を信号処理に限定してフィルタリング処理を高速に行えるアーキテクチャを有するＤＳＰ（Digital Signal Processor）[3] も１９８０年代初頭に誕生し，音声やモデムなどに適用され始めていた．しかし，映像信号を処理するには，毎秒数１０Ｍｂｙｔｅものデータを扱わなければならず，１個のデータに仮に１００ステップの処理を行うとしても数１０００ＭＯＰＳ（Million Operation per second）の処理能力が要求され，現状の機器の処理能力，処理機能との間に大きな落差があった．

そこで筆者は，標準ＴＶ，ＨＤＴＶ（High Definition TV），リモートセンシング画像などあらゆる画像・映像信号をプログラムにより処理できる汎用の画像・映像信号処理装置を作ることを目標に画像・映像信号用のリアルタイムプロセッサの研究に着手した．この概念を実現するために，まず小規模なマルチプロセッサ構成による実験システムＲＴＶＰ（Real-time video signal processor）を試作した[4]．このＲＴＶＰの試作により実用化への糸口を見いだし，得られた知見を踏まえて，映像信号用のＬＳＩ化プロセッサを開発し[5]，マルチプロセッサ構成による大規模な実用システムＰｉｃｏｔ－ｓｙｓｔｅｍ（Picture computer system）を開発した[6]．さらに，Ｐｉｃｏｔ－ｓｙｓｔｅｍを中核としたビデオスイッチャを実用化し[7]，ＮＨＫ放送センタの番組制作設備として運用している．

１．２　本論文の構成

本論文は，研究の背景と意義について述べた第１章に引き続いて，以下に示すように構成されている．

第２章では，放送用の映像信号処理の特質について考察を加える．特に，幾何学変換，効果波形発生，映像合成を取り上げ，放送用映像信号処理装置が具備すべき演算機能，演算精度について検討する．

第３章では，プログラム可能なビデオレート映像信号処理システムのアーキテクチャについて検討し，放送用の実時間映像信号処置装置の構成法について提案する．特に，並列処理方式，メモリ構成法，プロセッサ間同期方法，プログラムロード方式について取り上げて述べる．

第４章では，第２章，第３章の検討結果を踏まえて試作した実験システムＲＴＶＰのハードウェア，ソフトウェア構成について述べる．ＲＴＶＰは，プログラム可能なビデオレート映像信号処理システムの実現性の検証を行うために試作した実験システムであるが，ＲＴＶＰの処理能力などを評価し，実用化する上での問題点，改善すべき点などについて述べる．

第５章では，実験システムＲＴＶＰの試作によって得られた知見に基づき，開発した実用システムＰｉｃｏｔ－ｓｙｓｔｅｍについて述べる．Ｐｉｃｏｔ－ｓｙｓｔｅｍは，１４．３ＭＨｚの速度で複数のカラー動画像をビデオレート処理可能なマルチプロセッサであるが，小型化のためにプロセッサＬＳＩとネットワークＬＳＩの２種のＬＳＩを開発しており，第５章ではこれらのＬＳＩも含めハードウェア，ソフトウェア構成について述べる．また，実用化した各種応用例として，ビデオスイッチャ，画質補正，動き検出，映像合成などについても言及する．

第６章では，研究成果についてまとめ，今後の研究課題について述べることにより結言とする．

［２］映像信号処理に関する検討

２．１　映像信号のディジタル処理

放送における映像信号処理は，従来アナログで行われてきたが，１９７２年のディジタルＴＶ方式変換装置を契機にディジタル方式への移行が始まった．当初は，アナログ回路では実現できない機能をディジタルで行うといった方法が採られていたが，現在では，高画質，高安定などのメリットから，あらゆる装置のディジタル化が進行している．

元来，映像信号処理では，『１枚の画像を１／３０秒以内に途切れることなく処理する』リアルタイム性が必須であるため，アナログで１次元映像信号を直接処理していた．しかし，ディジタル素子技術の進歩から，高速にディジタル処理することが可能になり，さらに大容量メモリが安価に手にはいるようになり，２次元さらに３次元（２次元＋時間）で処理できるようになってきた．

ディジタル映像信号処理では，映像信号の標本化に当たって，コンポジット（輝度信号に色信号を重畳したＮＴＳＣ信号）で処理する場合と，コンポーネント（輝度信号と色信号が別になった[R,G,B]，[Y,R-Y,B-Y]，[Y,I,Q] などの信号形式）で処理する場合がある．

コンポジットで行う場合は，カラーサブキャリアの４倍の周波数である１４．３ＭＨｚの標本化周波数を採用するのが一般的である．ＮＴＳＣ信号の信号帯域が４．２ＭＨｚであるので，標本化周波数としては，標本化定理による必要最低限の標本化周波数の８．４ＭＨｚ以上あればよいが，

・Ａ／Ｄ，Ｄ／Ａの繰り返しによる劣化が少ない．
・プリフィルタ，ポストフィルタに特性の良いものが使える．
・サブキャリアによるビート妨害が目立たない．
・標本点が正方格子状に並び，ライン間，フレーム間の処理がやりやすい．
・カラーエンコード，デコードの処理がやりやすい．

などの理由から通常１４．３ＭＨｚが使われている[8]．

また，コンポーネントの場合は，上記のような制約条件は少なく，信号帯域の２倍以上であればいくらでもよいが，通常はＣＣＩＲ勧告６０１[9] の１３．５ＭＨｚが使われている．

ＨＤＴＶ（High Definition TV，高精細度ＴＶ）については，世界中で各種の方式が提案されているが，日本ではＢＴＡ（放送技術協議会）規格[10] の走査線数１１２５本／６０フィールドのものが使われている．この場合は７４．２５ＭＨｚの標本化周波数が使用される．

表２．１に，本論文で主対象とする映像信号の規格を示す．表２．１下部

表２．１　映像信号規格と画素数

	NTSC		HDTV
走査線数	５２５本		１１２５本
フィールド周波数	５９．９４Ｈｚ		６０Ｈｚ
フレーム数	毎秒約３０コマ		毎秒３０コマ
信号帯域	４．２ＭＨｚ		３０．０ＭＨｚ
標本化周波数	１４．３ＭＨｚ	１３．５ＭＨｚ	７４．２５ＭＨｚ
全画素数	横９１０×縦５２５	横８５８×縦５２５	横２２００×縦１１２５
有効画素数	横７５４×縦４８３	横７２０×縦４８３	横１９２０×縦１０３５

の有効画素数のうち，１４．３ＭＨｚの項は，ディジタルに関しては規格が無いため日本の電波法に準じて算出した値である．

量子化数については，要求される画質に応じて８～１０ｂｉｔ位が選ばれる．最近は，１０ｂｉｔが好まれている．また，途中の処理は，１２ｂｉｔないし１６ｂｉｔで行われている[8]．

ディジタル映像信号処理機器には，ビデオスイッチャ，特殊効果装置，画質補償器，ＴＢＣ（Time Base Corrector），ＦＳ（Frame Synchronizer），カラーコレクタ，ノイズリデューサ，ＴＶ方式変換装置など様々な機器がある[11][12]が，これらを処理の観点からみると，表２．２に示す分類が考えられる．

表２．２　映像信号処理の分類

（ａ）処理の領域による分類

種類	処理内容	処理例
時空間	時間領域の処理 （走査が行われる映像信号では空間領域の処理が含まれる）	濃度値変換．映像混合．色補正など
周波数	周波数領域の処理 （通常，時空間領域の畳み込み演算にして行う）	ＹＣ分離．輪郭補償．帯域制限．デフォーカス．ノイズリデューサ．ＦＦＴなど
座標	座標軸を変更する処理 （透視変換．曲面マッピングでは浮動小数点演算が要求される）	拡大・縮小．回転．透視変換．２次曲面へのマッピングなどの幾何学変換
準時間	書き込みと読み出しのクロックが異なる処理 （標本化周波数以下のクロック制御が必要である）	時間的ゆらぎを補正するＴＢＣ，異種同期の映像信号を一致させるＦＳなど

表２．２（つづき）　映像信号処理の分類

（ｂ）入力数による分類

種類	処理内容	処理例
単一入力	１つの入力信号のみを使用する処理（同一画像内のライン間，フレーム間処理も含まれる）	濃度値変換，輪郭補償，ＹＣ分離，幾何学変換，ノイズリデューサ，色補正など
複数入力	２つ以上の入力信号を同時に使用する処理	映像合成（１０入力以上の場合もあり），カラーマトリックス（R,G,B－Y,R-Y,B-Y），ステレオ画像処理など

（ｃ）次元による分類

種類	処理内容	処理例
０次元	それ自身のデータしか使用しない（基本的には，メモリが不要）	濃度値変換，映像混合，コアリング色補正など
１次元	水平方向のデータを使用する（数画素のメモリが必要）	ＢＰＦによるＹＣ分離，水平輪郭補償，コアリングによるノイズ除去など
２次元	垂直方向のデータを使用する（数ラインのメモリが必要）	くし型フィルタ，垂直輪郭補償，ＴＢＣ，巡回型フィルクを使うデフォーカスなど
３次元	時間軸方向（ﾌｨｰﾙﾄﾞ/ﾌﾚｰﾑ間）のデータを使用する（ﾌｨｰﾙﾄﾞ/ﾌﾚｰﾑメモリが必要）	幾何学変換，３次元ＹＣ分離，動き検出，ＴＶ方式変換，ノイズリデューサなど

（ｄ）線形・非線形による分類

種類	処理内容	処理例
線形	線形演算要素による処理（積和演算で表現できるもの）	利得調整，輪郭補償，コンボリューション，フィルクなど
非線形	非線形演算要素による処理（ROMまたはRAMによるﾙｯｸｱｯﾌﾟﾃｰﾌﾞﾙが使える場合がある．）	γ補正，コアリング，白クリップ，メディアンフィルクなど

表２．２（つづき）　映像信号処理の分類

（ｃ）データ依存度による分類

種類	処理内容	処理例
非依存	濃度値，位置に関係なく全画面で均一の処理をする	濃度値変換，輪郭補償，コンボリューション，拡大・縮小，デフォーカスなど
濃度値依存	濃度値に依存して処理を変える	適応型ＹＣ分離，動き補償型ＴＶ方式変換器，動き補償型ノイズリデューサ，輪郭トラッキングなど
位置依存	画面上の位置に依存して処理を変える	シェーディング補正，レジストレーション補正，透視変換用の折り返し除去フィルタなど

以上の分類から，処理に必要な回路，例えばメモリの有無，入力数などについて知見が得られるが，以下では，幾何学変換，波形発生，映像合成を取り上げ，処理に必要な精度，機能，回路などについて詳しく検討する．

２．２　幾何学変換

２．２．１　幾何学変換回路

従来からリモートセンシング画像など各種の画像処理において，物体の位置合わせ，レンズを含めた撮像系の幾何学歪みの補正，画像の重ね合わせなどの幾何学変換が行われてきた．一方，放送分野においても，番組制作の中で，映像表現のための特殊効果の一つとして，拡大縮小をはじめとする幾何学変換が頻繁に行われている．

この様な幾何学変換では，（ａ）２次元空間上で標本化された各画素の位置を変更するアドレス変換処理，（ｂ）標本点の間隔が原画像に対して粗くなることによって生ずる折り返し歪を帯域制限して除去するための前置フィルタ処理，（ｃ）変換すべき位置に標本点が存在しないときに近傍の画素から内挿して求めるための補間フィルタ処理の主な３つの処理が必要である．幾何学変換には，拡大・縮小，平行移動などの線形な変換から，透視変換，２次以上の曲面へのマッピングなどの様々な変形があるが，これらを一画素毎に正確に計算すると，多くの時間を要しハードウェアの規模も大きくなる．実現性を重視すると，できるだけ小さいハードウェアで十分な画質を維持することが重要となる．

以下では幾何学変換回路の構成法について検討する．幾何学変換回路のうち，前置フィルタ処理に関しては他にも検討されている[13][14]ので，ここ

では，アドレス変換処理，補間処理について検討する．

2．2．2　アドレス変換処理

幾何学変換の中でも最も一般的で使用頻度の高いアフィン変換を取り上げ，アドレス演算のための回路構成と演算語長について検討する．

２次元のアフィン変換は，

$$x = A \cdot X + B \cdot Y + C \qquad (2.1)$$

$$y = D \cdot X + E \cdot Y + F \qquad (2.2)$$

で示すことができる．ここで，(x,y) は変換前の座標，(X,Y) は変換後の座標，A,B,C,D,E,F は，変換パラメータである．この変換は，乗算器を用いて忠実に実現する方法もあるが，図２．１に示すように加算器のみで構成できる．すなわち，走査に合わせて，まず画面左上では演算値レジスタhacc1，vacc1 にそれぞれ初期値レジスタhint1,vint1 から C,Fの初期値を与える．水平方向の走査に合わせ，A が格納されている増分値レジスタhinc1，Dが格納されている増分値レジスタvinc1 を毎クロック加算する．１ラインの処理が終ったら，Yを1増やしX＝0として，次のラインの処理を開始する．すなわち，B が格納されている増分値レジスタhinc2，Eが格納されている増分値レジスタvinc2 を加える．しかし，X＝0とするので，B,E を単純に加算してはならない．このために演算値レジスタとしてhacc2,vacc2 を用意し，X＝0のときのx,y の値を保存しておき，次のラインに移動した際にhacc2,vacc2 からそれぞれhacc1,vacc1 に戻して，B,E を加算する．この方法により，（２．１）（２．２）式の計算が加算のみで実現できる[15]．

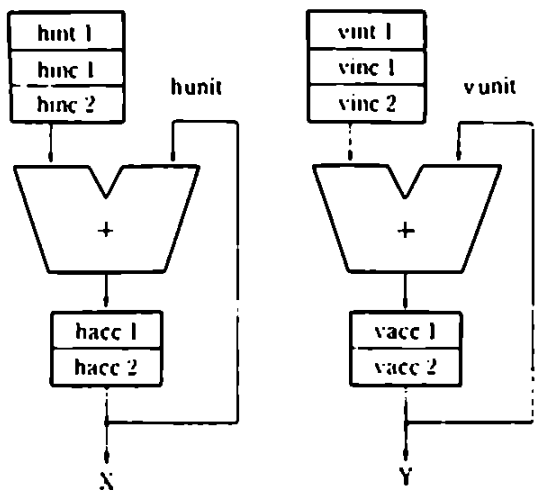


図２．１　アフィン変換回路

次に，拡大・縮小と回転に分け，演算語長について検討する．なお，アフィン変換には，この他に移動とせん断があるが，回路構成，演算語長については，拡大・縮小，回転と同様なので省略する．

2．2．3　拡大・縮小における演算語長

拡大・縮小は，（2．1）（2．2）式において，B＝C＝D＝F＝0 の場合であり，変換式は，

$$x = A \cdot X \qquad (2.3)$$

$$y = E \cdot Y \qquad (2.4)$$

となる．横P縦Q画素の画像が，横p縦q画素に縮小された場合は，変換パラメータA,E は，

$$A = P/p \qquad (2.5)$$

$$E = Q/q \qquad (2.6)$$

である．このときに，図2．2(a)に示すように縮小されて，しかもその位置が画面の端のとき，反対側の端がオーバーフローしないで計算可能な演算語長が整数部の所要ビット数である．これを満たす条件は，

$$2^{m} > A \cdot P \qquad (2.7)$$

$$2^{n} > E \cdot Q \qquad (2.8)$$

である．ここで，m,n は，所要ビット数である．

（2．5）～（2．8）式を解くと，

$$m > 2 \cdot \log_2 (P) - \log_2 (p) \qquad (2.9)$$

$$n > 2 \cdot \log_2 (Q) - \log_2 (q) \qquad (2.10)$$

となる．

５２５本／６０Ｈｚの標準ＴＶを４倍のサブキャリアで標本化した場合（以下１４．３ＭＨｚと略す）を考えると，有効画素数は，表2．1によれば P＝754,Q＝483である．p＝1,q＝1すなわち全画素を１点に縮小しても，オー

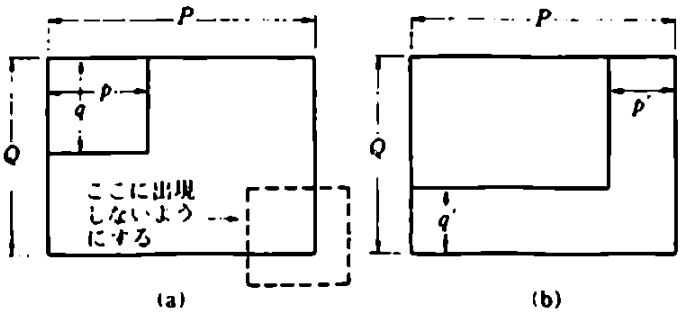


図２．２　拡大・縮小

表２．３　拡大・縮小に必要なビット数（整数部）

（単位：ビット）

縮小サイズ p, q 画素	標準テレビ				HDTV	
	14.3 MHz		13.5 MHz		74.25 MHz	
	m	n	m	n	m	n
1/4	21.1	19.8	21.0	19.8	23.8	22.0
1/2	20.1	18.8	20.0	18.8	22.8	21.0
1	19.1	17.8	19.0	17.8	21.8	20.0
2	18.1	16.8	18.0	16.8	20.8	19.0
4	17.1	15.8	17.0	15.8	19.8	18.0
8	16.1	14.8	16.0	14.8	18.8	17.0
10	15.8	14.5	15.7	14.5	18.5	16.7
20	14.8	13.5	14.7	13.5	17.5	15.7

（注）　14.3 MHz のとき $P=754$，$Q=483$

13.5 MHz のとき $P=720$，$Q=483$

74.25 MHz のとき $P=1920$，$Q=1035$ としている．

表２．４　拡大・縮小に必要なビット数（小数部）

（単位：ビット）

縮小サイズ p', q' 画素	標準テレビ				HDTV	
	14.3 MHz		13.5 MHz		74.25 MHz	
	m	n	m	n	m	n
1/4	11.6	10.9	11.5	10.9	12.9	12.0
1/2	10.6	9.9	10.5	9.9	11.9	11.0
1	9.6	8.9	9.5	8.9	10.9	10.0
2	8.6	7.9	8.5	7.9	9.9	9.0
4	7.6	6.9	7.5	6.9	8.9	8.0
8	6.6	5.9	6.5	5.9	7.9	7.0
10	6.2	5.6	6.2	5.6	7.6	6.7
20	5.2	4.6	5.2	4.6	6.6	5.7

バーフローが生じないためのビット数を求めると，m＞19.1，n＞17.8となる．これが，整数部の所要ビット数である．その他の場合の所要ビット数を，ＣＣＩＲ勧告６０１（以下１３．５ＭＨｚと略す），ＢＴＡ規格ＨＤＴＶ（以下７４．２５ＭＨｚと略す）の標本化周波数の場合も含め，表２．３に示す．

一方，小数部の精度については，１倍に限りなく近い拡大・縮小ができるかどうかで決まる．図２．２(b)に示すように水平p'画素，垂直q'画素の縮小を想定すると，

$$1/2^{m'} < p'/P \qquad (2.11)$$

$$1/2^{n'} < q'/Q \qquad (2.12)$$

を満たすm'，n'が小数部に必要なビット数である．（２．11）（２．12）式を変形すると，

$$m' > \log_2(P) - \log_2(p') \qquad (2.13)$$

$$n' > \log_2(Q) - \log_2(q') \qquad (2.14)$$

になる．P＝754，Q＝483，p'＝1，q'＝1の場合は，m'＞9.6，n'＞8.9必要である．その他の場合の所要ビット数を表２．４に示す．

２．２．４　回転における演算語長

図２．３に示す回転は，

$$x = X\cdot\cos\theta - Y\cdot\sin\theta \qquad (2.15)$$

$$y = X\cdot\sin\theta + Y\cdot\cos\theta \qquad (2.16)$$

で示すことができる．ここで，θは回転角である．係数cosθ，sinθを表現するビット長が，回転の精度を決定する．

図２．１の回路構成を想定して，sinθ，cosθ値を表現するビット数に制限を加え，0～180°の範囲で回転させて行ったシミュレーション結果が表２．５である．表２．５は，本来の位置（理論値）からずれた距離の最大値を示す．これによれば，１４．３ＭＨｚの場合，蓄積誤差を１画素以内に抑える

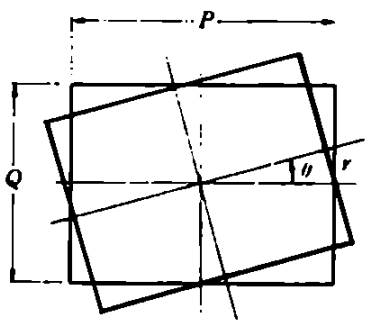


図２．３　回転

表２．５　回転におけるビット数と演算誤差

（単位：画素）

ビット数	標準テレビ		HDTV
	14.3 MHz	13.5 MHz	74.25 MHz
1	620.3	600.6	1510.1
2	303.4	293.8	738.8
4	77.3	74.8	188.0
6	19.5	18.9	47.5
8	4.7	4.5	11.3
10	1.2	1.1	2.9
11	0.6	0.6	1.4
12	0.3	0.3	0.7
14	0.1	0.1	0.2
16	0.0	0.0	0.0

ためには，小数部に１１ビット必要であることが判る．

２．２．５　アフィン変換における演算語長

以上を総合すると，アフィン変換の演算において必要な演算語長は，放送用途で最低限必要な１画素単位の精度を要求すると，１４．３ＭＨｚの場合３１（整数部２０，小数部１１）ビット，１３．５ＭＨｚの場合３０（整数部１９，小数部１１）ビット，７４．２５ＭＨｚの場合３４（整数部２２，小数部１２）ビット必要である．しかし，これは小数点位置を固定した場合で，拡大率に応じて小数点の位置を変える方法をとれば，演算語長を小さくできる．１４．３ＭＨｚを例に取り上げると，拡大率が大きい場合は，上記検討結果より整数部２０ビットあれば十分で，この場合は小数部の精度はあまり必要でない．一方，拡大率が１に近い場合は，上記検討結果から小数部に１１ビットあればよく，整数部は移動を考慮しても１１ビットあればよい．従って，１４．３ＭＨｚの場合は，１画素までの精度を要求しても計２２ビットあれば十分である．

２．２．６　補間処理

前節で述べた（２．１）（２．２）式の幾何学変換で求めた座標(x,y)は，変換前の座標系において必ずしも格子点に一致しないため，図２．４に示す

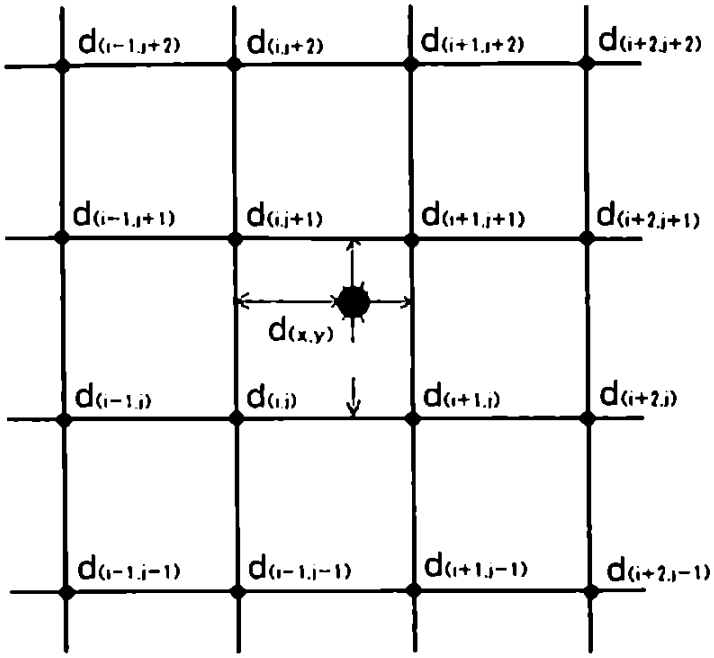


図２．４　補間

近傍の格子点を使って補間する必要がある．補間には，以下に示すような各種の方式がある[16][17]．

（ａ）最近傍零次補間（ＮＥＡＲ）

座標(x,y) に距離的に最も近い格子点を割り当てる方法である．すなわち，座標(x,y) の濃度値をd(x,y)とすると，

$$d(x,y)=d(i,j)$$
$$i=[x+0.5],\quad j=[y+0.5] \qquad (2.17)$$

とするものである．ここで，[ ] は括弧内の数値を超えない整数値を表すガウス記号である．

（ｂ）最大値零次補間（ＭＡＸ）

座標(x,y) を囲む近傍４点の画素のうち最も信号レベルが高いものを補間データとする方法である．式で示すと，

$$d(x,y)=MAX[\ d(i,j).d(i+1,j).d(i,j+1).d(i+1,j+1)\ ]$$
$$i=[x],\quad j=[y] \qquad (2.18)$$

である．ここで，MAX[ ]は括弧内の最大値のものをとる．

（ｃ）４点線形補間（４Ｐ）

近傍４点から線形に内挿して求める方法である．式で示すと，

$$d(x,y)=[f_1(y)\ \ f_2(y)]\begin{bmatrix} d(i,j) & d(i+1,j) \\ d(i,j+1) & d(i+1,j+1) \end{bmatrix}\begin{bmatrix} f_1(x) \\ f_2(x) \end{bmatrix}$$

$i=[x]$, $j=[y]$

$f_1(t)=1-(t-[t])$, $f_2(t)=t-[t]$, $f_1(t)=1-f_2(t)$ （２．19）

である.

（ｄ）９点２次補間（９Ｐ）

近傍９点から２次多項式により補間を行うもので，ラグランジュの補間公式から補間係数を求める．式で示すと，

$$d(x,y)=[f_1(y)\ \ f_2(y)\ \ f_3(y)]\begin{bmatrix} d(i-1,j-1) & d(i,j-1) & d(i+1,j-1) \\ d(i-1,j) & d(i,j) & d(i+1,j) \\ d(i-1,j+1) & d(i,j+1) & d(i+1,j+1) \end{bmatrix}\begin{bmatrix} f_1(x) \\ f_2(x) \\ f_3(x) \end{bmatrix}$$

$i=[x]$, $j=[y]$

$f_1(t)=(t-[t])(t-[t]-1)/2$

$f_2(t)=1-(t-[t])^2$

$f_3(t)=(t-[t])(t-[t]+1)/2$ （２．20）

である.

（ｅ）１６点標本化関数補間（ＳＩＮＣ）

標本化関数 sin(x)/x を用いると，周波数上で理想的な補間が行えるが，無限の標本点による畳み込みが必要になり現実的でない．そこで，近傍１６点を用いた補間を考える．すなわち，

$$d(x,y)=[f(y_1)\ \ f(y_2)\ \ f(y_3)\ \ f(y_4)]\begin{bmatrix} d(i-1,j-1) & d(i,j-1) & d(i+1,j-1) & d(i+2,j-1) \\ d(i-1,j) & d(i,j) & d(i+1,j) & d(i+2,j) \\ d(i-1,j+1) & d(i,j+1) & d(i+1,j+1) & d(i+2,j+1) \\ d(i-1,j+2) & d(i,j+2) & d(i+1,j+2) & d(i+2,j+2) \end{bmatrix}\begin{bmatrix} f(x_1) \\ f(x_2) \\ f(x_3) \\ f(x_4) \end{bmatrix}$$

$i=[x]$, $j=[y]$

$f(t)=(\sin\pi t)/\pi t$

$u=x-[x]$, $v=y-[y]$

$x_1=1+u$, $x_2=u$, $x_3=1-u$, $x_4=2-u$

$y_1=1+v$, $y_2=v$, $y_3=1-v$, $y_4=2-v$ （２．21）

を用いる.

（ｆ）３次スプライン補間（ＳＰ）

コンピュータグラフィックス等では，Ｂ－スプライン補間がよく用いられているが，補間曲線が端点を除いて標本点を通らないため，今回の適用例のように４標本点内の中間の曲線を求める場合，標本点に近い部分での誤差が大きくなる．このため，３次スプライン関数を用いる．すなわち，点t_1,t_2,t_3,t_4（$t_1<t_2<t_3<t_4$）に対して，関数値$f(t_1)$,$f(t_2)$,$f(t_3)$,$f(t_4)$が与えられており，各区間$[t_i,t_{i+1}]$（i=1,2,3,4）において３次の多項式で，１次導関数$f'(t)$および２次導関数$f''(t)$が全区間で連続で，$S(t_i)=f(t_i)$（i=1,2,3,4）である関数$S(t)$が，３次スプライン関数である．ここで，格子間隔が一定（＝１）で，端条件として$f'(t_1)$,$f'(t_4)$を与えられた４標本点が一つの３次多項式を通る条件で，区間$[t_2,t_3]$での補間係数を求め，この係数を使って補間値を求める．水平方向４標本点から補間を求めた後，垂直方向に並ぶ４点の標本点から再度スプライン補間により最終結果の補間値を得る．

上記６種の補間フィルタの周波数振幅特性を，図２．５に，補間誤差を図２．６に示す．図２．５は，１次元信号を３倍に拡大する場合について計算したものである．最近傍，最大値補間の特性は零次ホールドとみなして求めた．スプライン補間は，入力信号の標本値によって補間係数が変化するので省略した．図２．６は，１次元の正弦波信号を拡大し各種補間法で補間した波形と，拡大した時に得られる周波数で作った理想的な正弦波との２乗誤差を，周波数ごとに計算したものである．具体的には，$\cos\omega n$（n＝1,2,…，

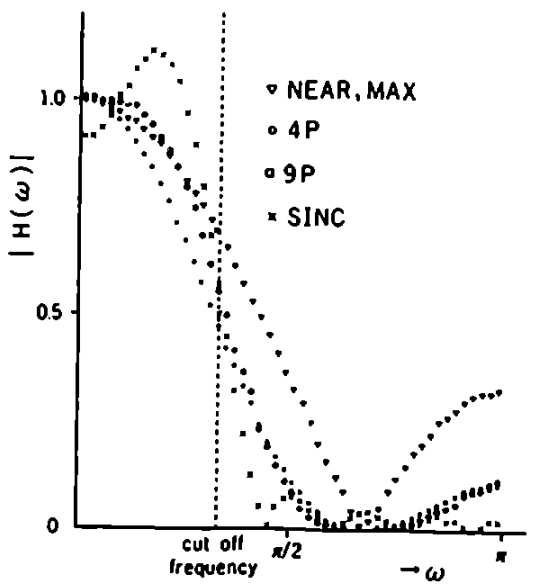


図２．５　補間関数の周波数振幅特性

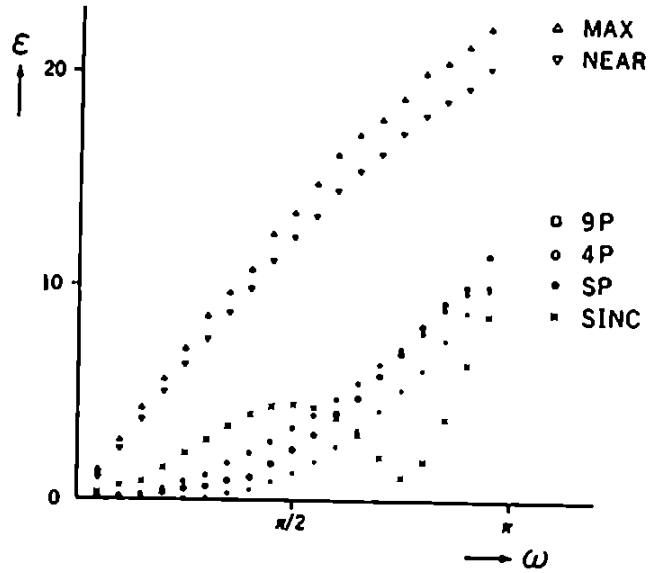


図２．６　補間関数の補間誤差

512）の波形を時間軸でｋ倍に拡大し補間して得られた信号をs(n)とし，拡大率に応じてω/kにした周波数の正弦波信号をg(n)とすると，

$$\varepsilon = \sqrt{\sum_{n=1}^{511} \{s(n)-g(n)\}^2} \qquad (2.\ 22)$$

を誤差の評価関数とした．図２．６は，k＝3.5の場合の例である．その他の拡大率についてもほぼ同様な傾向の誤差量を持つ．

２．２．７　補間処理の画質評価

上述の補間法と画質との関係について調べるために主観評価実験を行った[18]．実験は，表２．６に示すように，各種幾何学変換における補間と画質の関係を調べる実験１と，各種拡大率における補間法と画質の関係を調べる実験２の２種類行った．この実験では，テレビジョンシステム評価用ディジタル標準画像（ＴＶ学会）[19]のうち，細かい絵柄や直線の多い”チューリップ”と比較的平坦で曲線も含む“美人バタン”の２種類を素材に，各種の幾何学変換を施した画像を用いて，主観評価実験を行った．評価画像は，横７５４×縦４８３画素の画像で，ミニコンクラスのコンピュータ（ＭＶ４０００）を用いて浮動小数点演算（仮数部２４ビット，指数部８ビット）で計算した．変換画像の中で縮小される部分があり，正確には折り返し歪防止用のプリフィルタが必要であるが，今回の実験が補間法を評価するためのもので，①他の要因による影響を排除するためと，②縮小率が大きくないため，

表２．６　補間方式に関する主観評価実験

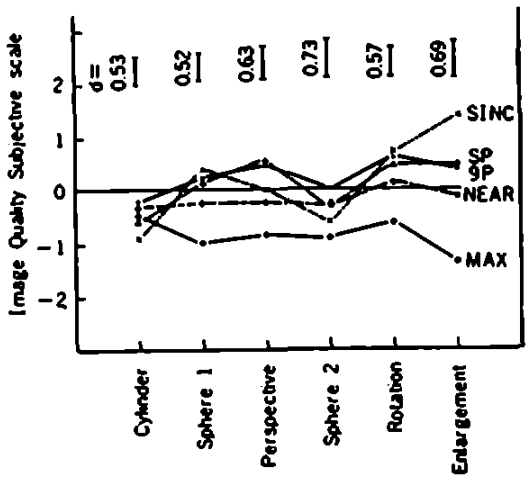
実験１　各種幾何学変換における補間と画質の関係を調べる．

評価法	４点補間を基準とした一対比較法 評価カテゴリ（７段階） ３：非常に良い ２：良い １：やや良い ０：同じ －１：やや悪い －２：悪い －３：非常に悪い
評価画像	"チューリップ"と"美人バタン"に対し， 回転（画面中心で４５°の回転）， 拡大（線分比で１．５倍の拡大）， 円筒（画面高の約１／２の直径と高さをもつ円柱にマッピング） パース（画面左部で約２倍，画面右部で１／２に拡大）， 球体１（画面高の約１／２の直径の球体にマッピング）， 球体２（画面高の約３／４の直径の球体にマッピング） の６種の幾何学変換を施した画像
評定者	放送技術者１３名
視距離	４Ｈ（画面高の４倍）
表示条件	カラーモニタ（２０インチ） 輝度：最大２００nit　最小２０nit モニタ背景：灰色３０nit

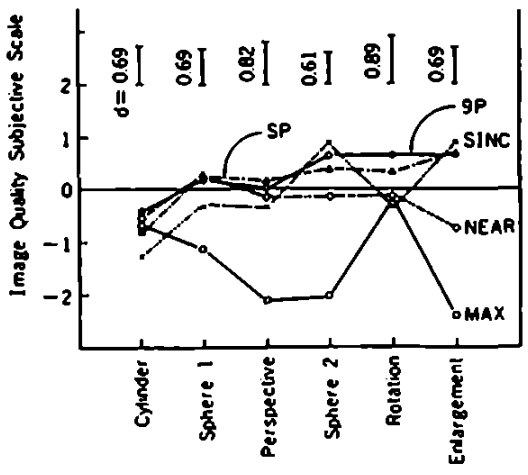
実験２　各種拡大率における補間法と画質の関係を調べる．

評価法	一対比較法 評価カテゴリ（７段階） ３：非常に良い ２：良い １：やや良い ０：同じ －１：やや悪い －２：悪い －３：非常に悪い
評価画像	"チューリップ"と"美人バタン"に対し， 0.7倍，1.5倍，2.0倍，2.7倍，3.5倍の６種の拡大率で 拡大・縮小を施した画像
評定者	放送技術者１３名
視距離	４Ｈ（画面高の４倍）
表示条件	カラーモニタ（２０インチ） 輝度：最大２００nit　最小２０nit モニタ背景：灰色３０nit

入れていない．実験結果を図２．７，図２．８に示す．図２．７は，各種幾何学変換における補間と画質の関係を示すもので，図２．８は，各種拡大率における補間法と画質の関係を示すものである．（ａ）が“美人バタン”，（ｂ）が“チューリップ”の場合である．縦軸は各評価画像が得た評価カテゴリの平均評点である．系列範ちゅう法により尺度化しており，図の縦軸は画質評価心理尺度である．各幾何学処理における二つの補間法の間隔が，図中に示すｄの距離より小さければ，その間には危険率５％で画質の有意差がないことを表す．

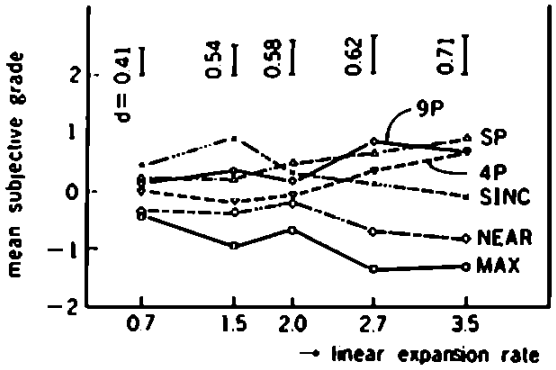


(a) "Girl"

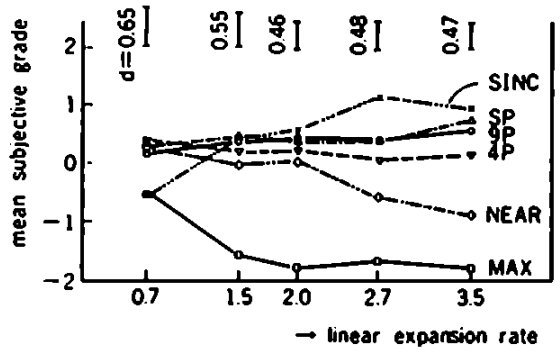


(b) "Tulips"

図２．７　主観評価実験１の結果



(a) "Girl"



(b) "Tulips"

図２．８　主観評価実験２の結果

図２．７，図２．８によれば，最近傍，最大値補間は，他の補間法に比べ悪い評価であった．これは，図２．６において補間誤差が他の補間法に比べ大きいこととも一致している．９点２次補間，スプライン補間法は全体的に良い結果を得ている．標本化間数補間は，評価が良い時と悪い時がある．標本化間数補間の周波数特性は，図２．５によれば高域が持ち上がっており，他の補間法に比して鮮鋭に見えるため，細かい画像のとき良い結果を与える．その反面，ノイズ成分も強調するため，背景の平坦な部分でノイズ（畳じわのような妨害）が目立ち，評価が悪くなる場合もあったものと考えられる．また，４点線形補間は，１．５倍および２倍拡大の“美人パタン”と３．５倍拡大の“チューリップ”を除くと９点２次補間，スプライン補間との間に有意差はなく，ほぼ良い結果を得ている．

各種補間法のハードウェア化に際して回路規模に大きな影響を与える処理の複雑さを，ミニコンピュータ（ＭＶ４０００）の計算時間で評価すると表２．７のようになった．表２．７は，４点線形補間を１としたときの相対時

表２．７　各種補間法のＣＰＵ時間

最近傍補間	０．５
最大値補間	０．６
４点線形補間	１．０
９点２次補間	１．８
１６点標本化補間	６．６
３次スプライン補間	８．２

（注）４点線形補間を１とした相対値
拡大のためのアドレス演算も含む

間で表している．主観評価実験と演算量から評価すると，４点線形補間は，計算量が少ない（３次スプライン補間の約１／８，９点２次補間の約１／２）わりに画質が良く，ハードウェア化に適している．

２．２．８　補間処理の演算語長

以下では，上記実験・検討結果に基づき，４点線形補間を取り上げ，必要な演算精度について検討する．（２．19）式を展開すると，

$$d(x,y) = (1-q)\cdot[(1-p)\cdot d(i,j) + p\cdot d(i+1,j)] + q\cdot[(1-p)\cdot d(i,j+1) + p\cdot d(i+1,j+1)] \quad (2.23)$$

となる．ここで，$p = f_2(x)$，$q = f_2(y)$である．

以下では，この補間に必要なビット数について検討する．p,q の演算語長を短くした場合に倍精度の浮動小数点で補間した場合に比べ，誤差が大きくなるのは，近傍４点の値が表現できる濃度値の最大値(max) と最小値(min) になった場合である．この組合せは，

① $d(i,j) = d(i,j+1) = max$，$d(i+1,j) = d(i+1,j+1) = min$　（２．24）

② $d(i,j) = d(i,j+1) = d(i+1,j) = max$，$d(i+1,j+1) = min$　（２．25）

③ $d(i,j) = max$，$d(i,j+1) = d(i+1,j) = d(i+1,j+1) = min$　（２．26）

④ $d(i,j) = d(i+1,j+1) = max$，$d(i,j+1) = d(i+1,j) = min$　（２．27）

と，これらを９０°，１８０°，２７０°回転させたパタンである．濃度レベル８ビットの場合max＝255，min＝0で，その差は２５５量子化レベルとなるはずである．しかし，ＮＴＳＣ信号の帯域は，１４．３ＭＨｚ標本化の場合の伝送帯域の７．１５ＭＨｚよりも４．２ＭＨｚと低く，隣合う画素の濃度レベル差は２５５量子化レベルにならない．１４．３ＭＨｚ標本化の場合に

4．２ＭＨｚの理想ローパスフィルタを通して計算すると，最大濃度レベル差は３８量子化レベルである．同様に計算すると，１３．５ＭＨｚの場合４０量子化レベル，７４．２５ＭＨｚの場合５２量子化レベルである．それぞれの場合の誤差を計算したものが，表２．８である．表中の平均誤差は，p，q を０～１の範囲で変化させた時に生ずる誤差の平均を示し，最大誤差は最も大きい誤差を示す．この誤差は，幾何学変換においてはブロック歪となり，非常に目立つ．これについては，Rico，Piper，Pieron の報告[20]があり，視覚的に敏感であることが知られている．また，視覚特性の検知限では，偽輪郭では３量子化レベル，パタン歪では１量化レベルに誤差を抑える必要があることも報告されている[21]．この条件を満たすためには，補間ビットとして平均誤差で５ビット，最大誤差で７ビット以上必要である．

ここで，ブロック歪について検討してみよう．横Ｍ倍，縦Ｎ倍の拡大率，ｎビット補間として，モザイク状にならない条件，すなわち同一の濃度レベルの画素が複数個生じない条件を求めると，

$$M<2^n,\quad N<2^n \tag{2．28}$$

である．ｎについて求めると，すなわち，

$$n>\log_2(M),\quad n>\log_2(N) \tag{2．29}$$

表２．８　補間ビット数と誤差

（単位：量子化ビット）

補間ビット数	標準テレビ				HDTV	
	14.3 MHz		13.5 MHz		74.25 MHz	
	最大	平均	最大	平均	最大	平均
1	28.5	8.0	30.0	8.4	38.9	10.9
2	16.6	4.0	17.4	4.2	22.7	5.5
3	8.8	2.0	9.3	2.1	12.1	2.7
4	4.6	1.0	4.8	1.3	6.2	1.4
5	2.3	0.5	2.4	0.4	3.2	0.7
6	1.1	0.2	1.2	0.2	1.6	0.3
7	0.6	0.1	0.6	0.1	0.8	0.2
8	0.3	0.1	0.3	0.1	0.4	0.1
9	0.1	0.0	0.1	0.0	0.2	0.0
10	0.1	0.0	0.1	0.0	0.1	0.0
11	0.0	0.0	0.0	0.0	0.0	0.0

となる．１画素を全画面に拡大した場合を想定すると，１４．３ＭＨｚの場合，M＝754,N＝483 であり，ｎは１０ビット以上必要である．しかし，これほどの拡大率の場合はブロック歪を除去したとしても，ボケが激しく実用に耐えうる画質ではない．筆者らの経験では，せいぜい１６倍（面積で２５６倍）が限界である．１６倍以下に限定すれば４ビットの精度でよく，上記の条件よりも緩いので問題ない．

以上を総合すると，１量子化レベルの誤差とするためには，平均誤差で５ビット，最大誤差で７ビット必要になる．

２．３　波形発生

２．３．１　ビデオスイッチャ

ビデオスイッチャ[22]は，番組制作に関わる主な映像信号処理機能を統合化した装置で，多数の素材に対して，ディゾルブ，カット，ワイプ，スーパー，ＤＶＥ（Digital Video effect）などの各種の効果処理を行う．これらのなかで，シーンチェンジによく使用されるのが，カット，ディゾルブ，ワイプである．

ディゾルブは，入力映像をA(i,j)，B(i,j)，出力映像をM(i,j)とすると，

$$M(i,j)=k\cdot A(i,j)+(1-k)\cdot B(i,j) \qquad (2.30)$$

に示すように，混合比ｋを，フレーム毎に少しずつ０から１に変化させることにより，映像がＡからＢに混合されながら切り替える処理である．これに対して，カットは，ｋを０から１に１フレームで変化させることにより混合させることなく切り替える．

ワイプは，一方の画面が別の画面へ各種の図形パタンで，合成されながらその形が広がって取り切られて行く効果である．図２．９に示すように，キー信号がディゾルブやカットのように全画面で均一でなく図形パタンとなっており，（２．30）式のｋが画面上の位置に依存して変化する

$$M(i,j)=k(i,j)\cdot A(i,j)+(1-k(i,j))\cdot B(i,j) \qquad (2.31)$$

になっている．キー信号は，図２．１０(a)に示す波形が，(b)に示す特性を有するレベルスライス回路で２値化され，(c)に示す波形が発生される．キー信号には，境界部が急峻なハードキー（図２．１０中の実線）と，境界部が滑らかに変化するソフトキー（図２．１０中の点線）がある．

このワイプを行うための波形信号を作るのが効果波形発生回路であり，ワイプ以外にも幾何学変換を行う特殊効果装置などのマスキングにも使われている．

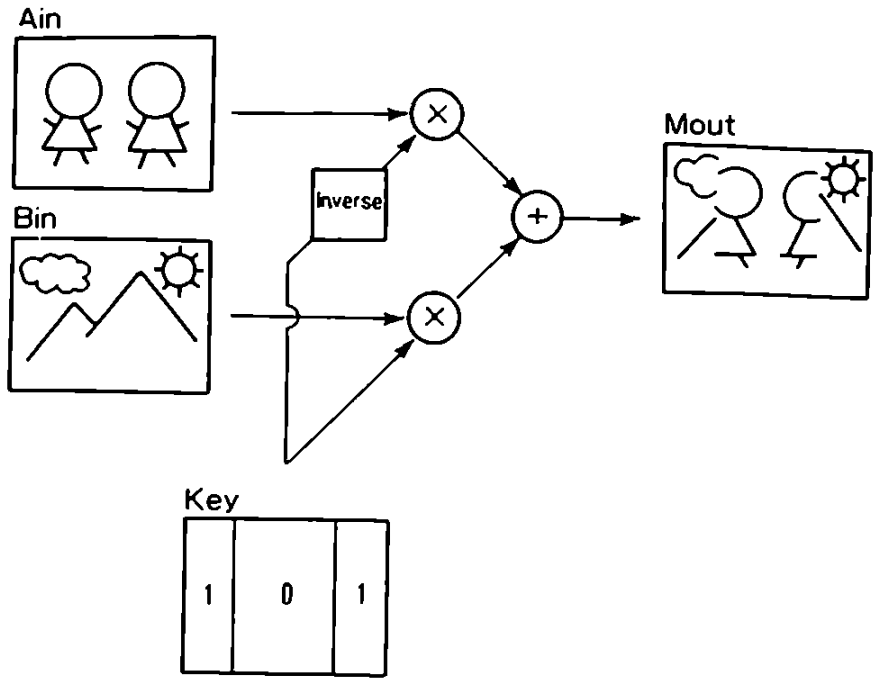


図２．９　ワイプ

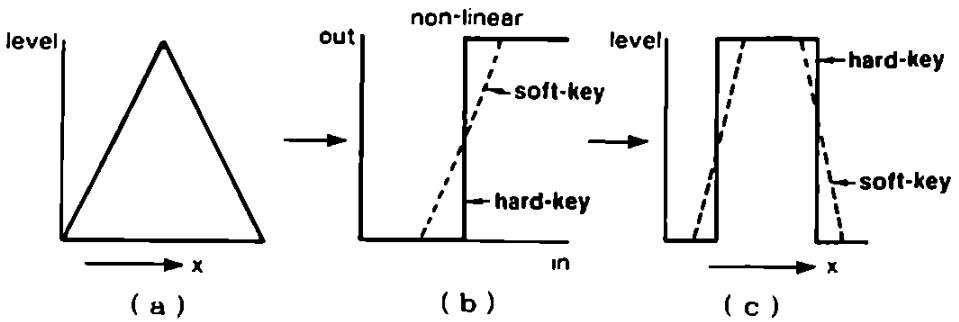


図２．１０　ワイプキー

２．３．２　効果波形発生回路

効果波形発生回路は，図２．１１に示す回路で構成できる[２３]．水平周期と垂直周期の三角波，のこぎり波などの基本波信号を元に，混合，スライスして，効果波形が生成される．図２．１１のスイッチ（Sw）を切り換えることにより，水平・垂直混合波形（Sw１側）及び独立波形（Sw２側）を発生することができる．

さて，基本波の発生方法であるが，三角波を例に取り上げて回路構成と演算語長について検討する．他の波形でも大差は無い．図２．１２(a)は，水平周期の三角波である．水平帰線期間が終了したら，水平周期Ｍの半分で最

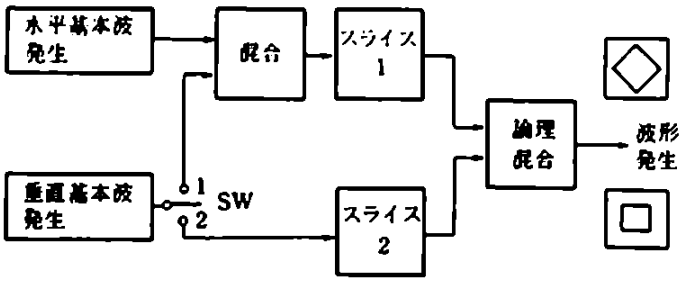


図２．１１　効果波形発生回路

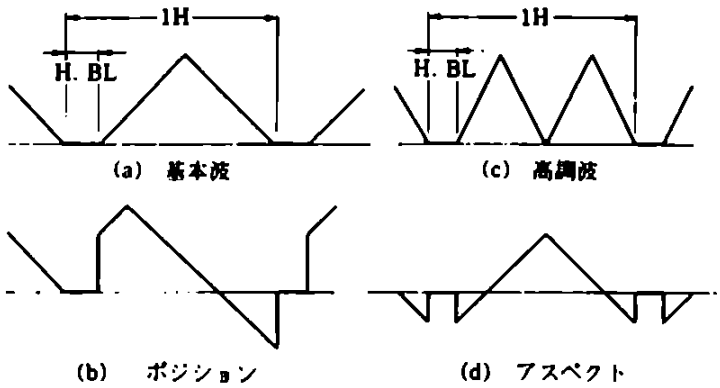


図２．１２　３角波

大値D になるように０から加算を行いライン中央になったら減算を行う．この動作を毎ライン行うことにより水平周期の基本波が発生できる．これを式で示すと，

$$f(x)=\begin{cases}\dfrac{2D}{M}x & [0 \leq x<\dfrac{M}{2}]\\ D-\dfrac{2D}{M}(x-\dfrac{M}{2}) & [\dfrac{M}{2} \leq x<M]\end{cases} \qquad (2.\ 32)$$

になる．ここで，x は水平アドレスであり，M は水平方向の有効画素数である．垂直方向の場合は，xをyに，MをN（垂直方向の有効画素数）に置き換えればよい．

上記波形を位置移動させる場合には，２．１２(b)に示す波形を発生させる．画面上のどの位置からでも波形を大きくして全画面をカバーして切り替わるようにするために，波形のダイナミックレンジをすべてカバーできるようなレベルスライスを行う必要がある．このときの波形は，

$$f(x)=\begin{cases}\dfrac{2D}{M}(x-p) & [0\leq x<\dfrac{M}{2}+p]\\ D-\dfrac{2D}{M}(x-p-\dfrac{M}{2}) & [\dfrac{M}{2}+p\leq x<M]\end{cases} \qquad (2.\ 33)$$

で示す式になる．ここで，p は移動量で $-M/2\leq p<M/2$ である．

図２．１２(c)の高調波は，増(減)分値を増やし波形を繰り返せばよい．m個の三角波がある場合を想定すると，

$$f(x)=\begin{cases}\dfrac{2mD}{M}(x-\dfrac{M}{m}i) & [\dfrac{M}{2m}2i\leq x<\dfrac{M}{2m}(2i+1)]\\ D-\dfrac{2mD}{M}(x-\dfrac{M}{2m}(2i+1)) & [\dfrac{M}{2m}(2i+1)\leq x<\dfrac{M}{2m}(2i+2)]\end{cases} \qquad (2.\ 34)$$

に示す式になる．ここで，$i=0.1.\cdots.m-1$である．

図２．１２(d)のアスペクトは，水平波形と垂直波形のバイアスを変えれば得られる．式で示すと，

$$f(x)=\begin{cases}\dfrac{2D}{M}x+D_0 & [0\leq x<\dfrac{M}{2}]\\ D-\dfrac{2D}{M}(x-\dfrac{M}{2})+D_0 & [\dfrac{M}{2}\leq x<M]\end{cases} \qquad (2.\ 35)$$

になる．ここで，D_0は，バイアスである．

これらの（２．32）～（２．35）式及びこれらの組み合わせを実行するには，図２．１のアフィン変換回路にコンパレータを追加すればよい．hunit でf(x)の計算を行い，vunit でx の条件計算を行う．コンパレータは，x の条件比較を行うために使用する．（x を比較対象としないで，f(x)そのものを最大値，最小値と比較して行う方法も可能である．）この比較は，演算と同時に毎クロック行う必要があるため，演算器と兼用することはできない．また，水平同期に対しては演算をリセットする必要がある．このために，プログラマブルな演算回路においては条件比較と水平同期の同時チェックができる機能が必要である．

垂直周期の基本波形発生の場合は，クロック毎ではなく，水平同期毎に加減算を行う．１走査線毎のワイプを実現するためには，第１フィールドと第２フィールドで初期値，最大値，最小値を変えなければならない．このためには，フレームパルスをチェックする機構が必要である．

ロータリー波形の場合は，基本波が，x,y の１価関数f(x).g(y) ではなく，

x,y の２価関数f(x,y),g(x,y) になる．f(x,y)を例に取り上げると，ライン毎の初期値がy に依存して増加する．（２．33）式のp がライン毎に変化することに相当する．

これらの演算において画素単位でワイプを可能にするためには，基本波形のダイナミックレンジとして水平及び垂直の画素数分の精度が最低限必要となる．従って，１４．３ＭＨｚ，１３．５ＭＨｚの場合１０ビット，７４．２５ＭＨｚの場合１１ビット必要である．

２．４　映像合成

２．４．１　クロマキーとビデオマット

ＴＶ番組制作における映像合成技術は，番組演出上，不可欠なものとなっており，より適用範囲の広い自然な合成手法が求められている．従来の映像合成法には，クロマキー[24]やビデオマット[25]などがある．クロマキーは，色情報を用いた物体抽出法で動物体の抽出が実時間で行えるが，背景に青色などの特殊なスクリーンが必要であり，適用できる映像は限定される．一方，対象物の輪郭を直接タブレットなどを用いて手動で指定してキー信号を作成するビデオマットでは，幅広い映像に適用可能であるが，動物体に適用するためには全てのフレームでキー信号を作成する必要があり，作業時間がかかる欠点がある．現在のところ，背景に制約を受けず，対象物自身が移動・変形する場合や，カメラがパン・ズーム・ドリーする場合に実時間で対応できる合成方法がない．

一方，コンピュータビジョンにおいては，動物体の３次元運動を解析する研究が行われている[26]．動画像の符号化においても，ズームなどの大局的な動き補償を用いた予測符号化の研究が行われている[27]．これらは映像合成を目的としておらず，テレビ番組で要求される種々雑多なシーンに適用可能な一般的な実時間処理手法は確立されていない．

以下では，動ベクトルを用いた映像合成法を提案し，本合成法に適した動ベクトル検出法について述べることにより，後述のシステム化に必要な機能について検討する．

２．４．２　提案する映像合成法の概要

図２．１３の原画像のような自然風景の中を走る車の背景を別の風景に変える合成を考える．本映像合成法では次に示す手順でキー信号を作成し，合成を行う．

STEP1　最初のフレーム上で対象物（図２．１３では，車）の輪郭線をマウスやタブレットなどを用いて手動で指示し，キー信号を作る．

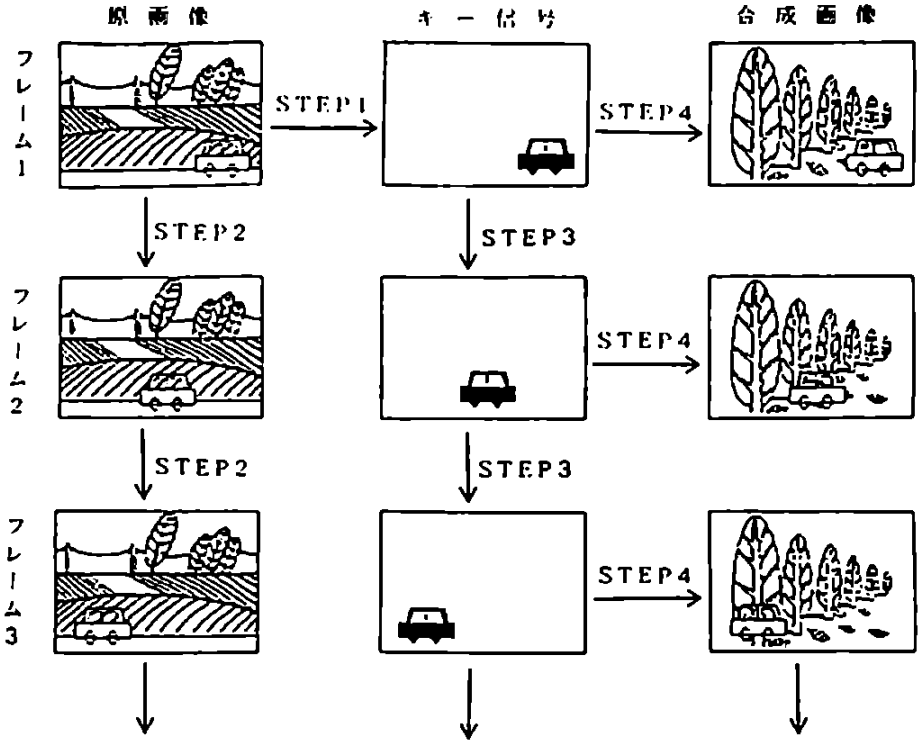


図２．１３　提案した映像合成法

STEP2　次のフレームとの間で，対象物の移動・変形などの動きを求める．
STEP3　求めた動きに合わせて現フレームのキー信号を移動・変形する．
STEP4　求めたキー信号を用いて対象物を抽出し，別の背景画像と合成する．
STEP5　STEP2.3.4 を繰り返し，連続した全てのフレームの合成を行う．

これらの手順のうち手動で行うのは STEP1のみで，残りは全て自動で行われる．

この映像合成法で問題となるのは，対象物の動きの求め方である．対象物の動きを求めるためには，現フレームの対象物上の各点が次のフレームのどこに対応するかをそれぞれ検出する必要がある．しかし，対象物上の各点の動きが全く独立である場合は少なく，対象物が，（１）大きさ・形を変えず平行移動している，（２）回転・ズームを伴って移動していると仮定しても，ＴＶ番組制作ではかなり多くのシーンに当てはまる．以下では，（１）（２）に分けて検討する．

２．４．３　動ベクトル検出法

（１）大きさ・形を変えず平行移動する場合

動物体が大きさ・形を変えず平行移動している場合，例えば物体がカメラ

の前を横切る場合や，カメラがパン・チルトする場合では，物体上の各点の動ベクトルは一様である．

従来より動ベクトル検出法として，（１）フーリエ変換を用いる方法，（２）相互相関関数を用いる方法，（３）勾配法，（４）マッチング法などが提案されている[24]が，実時間処理に向きかつ精度がよいとされるマッチング法を基本とし，動ベクトル検出を行うことにした．

マッチング法[25]は，現フレームS_nと前フレームS_{n-1}間で位置をずらしながら，類似度が最も高くなる位置を動ベクトルとする方法である．一般に，マッチング法では，画面を格子状に分割したブロック単位で検出を行い[26]，

$$P_n(x,y) = \sum_i |S_n(x_i+x, y_i+y) - S_{n-1}(x_i, y_i)| \qquad (2.\ 36)$$

で示される評価関数P_nが最小となる試行ベクトル(x,y)を動ベクトルとする．ここで，(x_i,y_i)はブロック内の画素の位置，Σはブロック内の全ての画素あるいは均一に配置された検出点についての総和を表す．マッチング法では，（１）ブロック内に特徴的な絵柄が無く単調な部分を多く含む場合は，候補となる動ベクトルが複数存在する場合があるため一意に決められない．（２）動物体と背景の両方を含む境界領域では，異なる動ベクトルをもつ代表画素の総和をとるために正確な検出ができないなどの問題がある．

そこで，本映像合成法では，抽出対象となる動物体（以下，対象物と呼ぶ）の形状が最初のフレームにおいて与えられていることを利用し，対象物上の画素をマッチングの対象領域とすることにした．さらに，演算量を減らすために対象物上の全点を用いてマッチングを行うのではなく，数ヶ所を検出点として選ぶようにした．検出点は対象物上のエッジ近辺の内側の離れた４エリア（それぞれ複数の画素を含む）に設定する．

これは，対象物上の内部では均一の明るさやテクスチャをもち，外部（背景）では異なる絵柄をもつ場合が多い点に注目したからである．図２．１４(a)に示すように，４角形の対象物が動ベクトルV_nだけ移動した場合，現フレームのエリア１は次フィールドのエリアＡと類似度が高く，（２．36）式のP_nは小さいはずである．マッチング法では探索範囲全てが候補領域であるが，対象物外部では異なる絵柄をもつ場合が多いという上記の仮定から，エリアＢなどの対象物内部では類似度が高くなる（P_nが小さくなる）可能性が大きい．対象物内部であるために，エリア１のP_nが小である確率が高い，すなわち動ベクトルとして選択できる領域を示すと，図２．１４(b)の ▨ の部分となる．エリア２，３，４についても，対象物内部であるために，動ベクトルとして選択できる可能性が高い領域を示すと，図２．１４(b)の ▨ となる．背景に対象物内部よりも類似したエリアがあれば誤検出

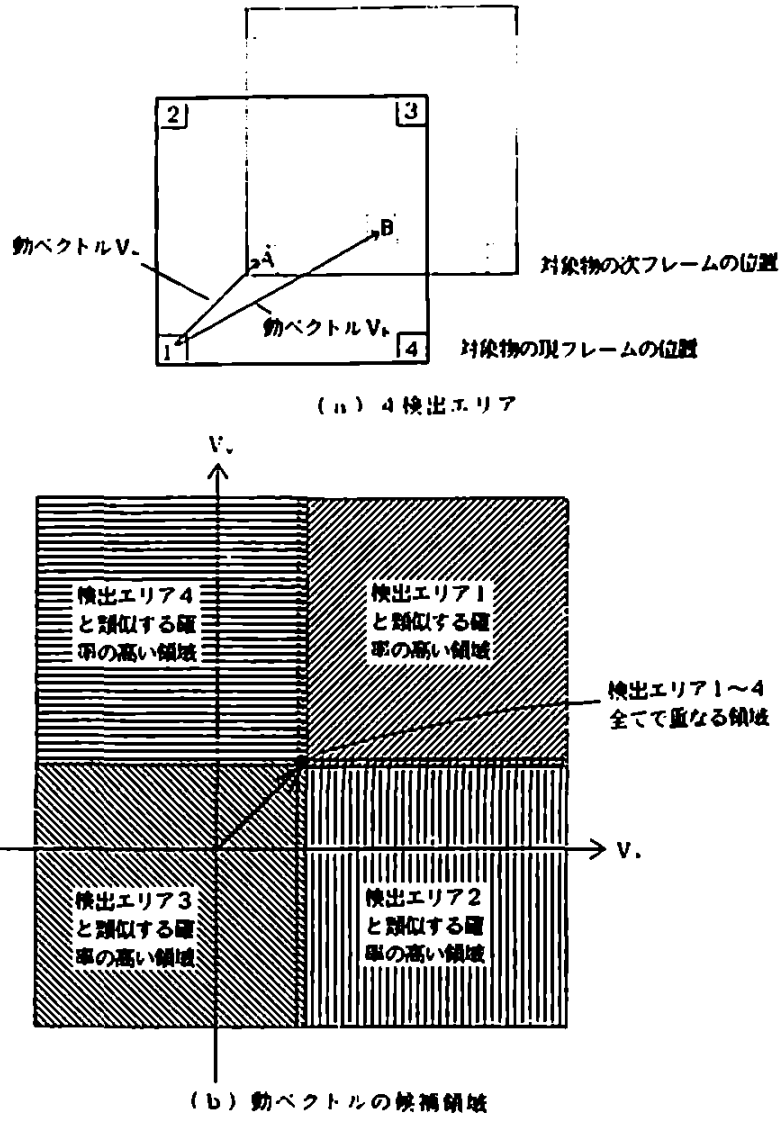


図２．１４　４検出エリアによる動ベクトル検出

の可能性もあるが，▨ ▨ ▨ 全てが重なる領域（４エリアとも同一の動ベクトルの値をとる領域）をとれば，真の動ベクトルになる確率が高いのでこれを動ベクトルとして採用する．検出点として演算時間の兼ね合いで４エリアを選んだが，より多くのエリアを設定すれば，誤検出の可能性は減る．

このようにして求めた動ベクトルを用いて検出点を移動させ，次のフレーム間に対しても同様にして検出を行う．以降，全てのフレームに対し検出点を移動させながら検出を続ける．

（２）拡大・縮小，回転を伴う場合

動物体が画面上で回転する場合や，カメラがズームする場合などでは，物

体上の各点の動ベクトルは一様ではない．この場合は，動物体が画面上で拡大・縮小および回転などの２次元アフィン変換に基づいて移動・変形していると考えられる．すなわち，対象物上の点(x_n, y_n)と，この点の次フレームでの対応点(x_{n+1}, y_{n+1})は２次元アフィン変換を用いて，

$$x_{n+1} = a \cdot x_n + b \cdot y_n + c \qquad (2.\ 37)$$

$$y_{n+1} = d \cdot x_n + e \cdot y_n + f \qquad (2.\ 38)$$

で表すことができる．ここで，a.b.c.d.e.f は，変換パラメータである．また，点(x_n, y_n)の対応点(x_{n+1}, y_{n+1})は，(x_n, y_n)の周辺の複数画素を使い，（２．36）式のマッチング法を用いて求める．３個の特異でない(x_n, y_n)と(x_{n+1}, y_{n+1})の対応（動ベクトル）が求まれば，６個の変換パラメータa.b.c.d.e.f を一意に決定できるが，検出精度を上げるために，多数の点を用いて最小二乗法[31]で求める．すなわち，k 個の対応点群 $(x_n(i), y_n(i)) \rightarrow (x_{n+1}(i), y_{n+1}(i))$ $(i=1,2,\cdots k)$ に対して，

$$E_x = \sum_{i=1}^{k} [(x_{n+1}(i) - a \cdot x_n(i) - b \cdot y_n(i) - c)^2] \qquad (2.\ 39)$$

$$E_y = \sum_{i=1}^{k} [(y_{n+1}(i) - d \cdot x_n(i) - e \cdot y_n(i) - f)^2] \qquad (2.\ 40)$$

が，最小となるa.b.c.d.e.f を求める．ここで，Σは，全ての対応点についての総和である．

このようにして求めた変換行列によって各検出点を移動させ，移動した点を新たな検出点として，次のフレーム間に対して同様に検出を行う．以降，全てのフレームに対し検出点を移動させながら検出を続ける．

２．４．４　シミュレーション実験

（１）大きさ・形を変えず平行移動する場合

静止画像の上に，静止物体をフレーム毎に位置を変えて合成しノイズを加えて作成した動ベクトルが既知の疑似的な動画像（２０フレームの白黒画像）を用い，動画像シミュレータ[32]上で実験を行った[33]．実験では，対象物の形状，明るさ，絵柄，動き，背景Ｓ／Ｎ，検出点の数，位置などのパラメータを変えて行った．実験結果の１例を図２．１５に示す．図２．１５は，６種類のテクスチャを有する対象物に対して，動き検出実験を行った結果で，検出点の数をパラメータにして，対象物の初期フレームからの移動量の理論値(x_r, y_r)と検出値(x_d, y_d)の累積移動量誤差$\sqrt{(x_d - x_r)^2 + (y_d - y_r)^2}$の最大値を示す．図２．１５によれば，検出点を対象物上のエッジ近辺の離れた４エリアに設定した場合，各エリア１５画素以上使用すれば，物体の明るさ，絵柄に影響されず，±２画素未満，すなわち，上下左右（以上誤差１画素），斜め（誤差$\sqrt{2}$画素）の８近傍点内の誤差で動き検出が行えることがわかる．

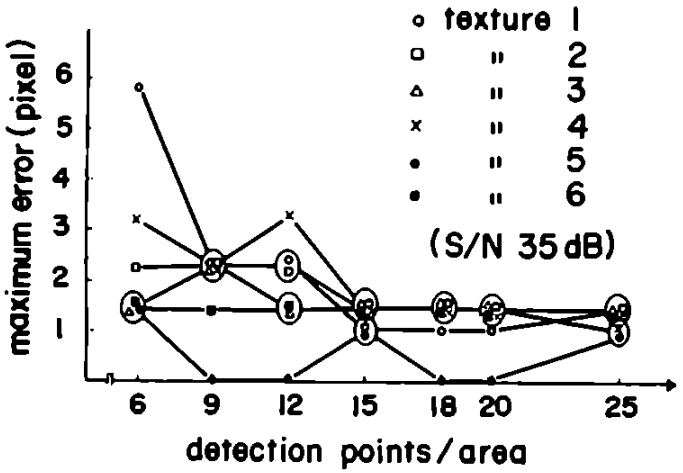


図２．１５　各検出エリア当りの画素数と検出誤差

次に，実際に撮影して得た動画像に対し，映像合成実験を行った[41]．実験は，固定したカメラの前を動物体が横切る画像（動画像１），カメラをパンさせて撮影した画像（動画像２），動物体が画面中央に位置するように追跡して撮影した画像（動画像３）の各々１０フレーム（２０フィールド）程度連続するカラー動画像に対して行った．使用したカラー動画像は，コンポーネント信号（R，G，B）で，動ベクトル検出の評価関数として，（２．36）式をコンポーネント・カラー画像に対応させた

$$P_n(x,y)=\sum\{|R_n(x_1+x,y_1+y)-R_{n-1}(x_1,y_1)|+|G_n(x_1+x,y_1+y)-G_{n-1}(x_1,y_1)|+|B_n(x_1+x,y_1+y)-B_{n-1}(X_1,Y_1)|\} \qquad (2.41)$$

を用いて動ベクトル検出を行った．ここで，R,G,B はＲＧＢ各コンポーネントの映像信号レベルを表す．図２．１６は１例で，(a)は初期フィールドに輪郭線を与えた画像，(b)は２０フィールド後の合成画像である．なお，合成は，合成の境界部でちらつきが見えなくなるように，２値のハードキーを５×５のウィンドウで平滑化して多値にしたソフトキーを用いて行った．図２．１６(a)で輪郭線上の○で示した部分が検出エリアである．図２．１７は，図２．１５と同様に検出エリア当りの画素数をパラメータとした場合の動ベクトルの検出結果である．最大誤差の計算は，図２．１５の場合の理論値(x_0,y_0)の代わりに，輪郭線近辺の全画素を用いて行った検出値(x_1,y_1)を基準値として行った．すなわち，それぞれの画素数で検出した検出値(x_n,y_n)との累積移動量誤差$\sqrt{(x_n-x_1)^2+(y_n-y_1)^2}$の最大値である．この$(x_1,y_1)$は，筆者が目視で検出した動きとも一致している．動ベクトル検出の評価

（ａ）撮影画像＋輪郭線＋検出点（○印）

（ｂ）合成画像

図２．１６　映像合成実験

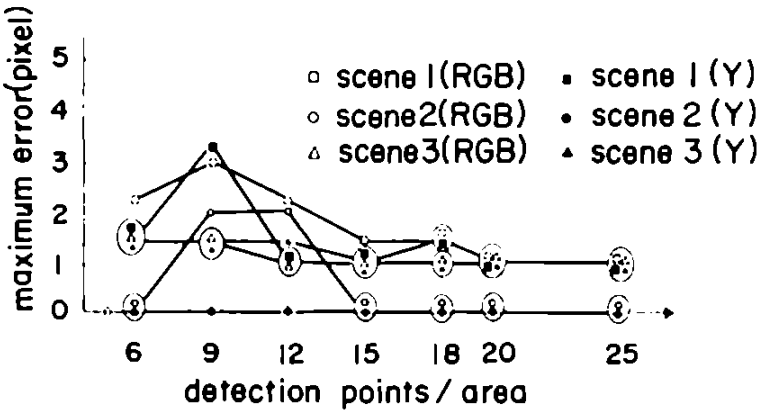


図２．１７　各検出エリア当りの画素数と検出誤差（撮影画像の場合）

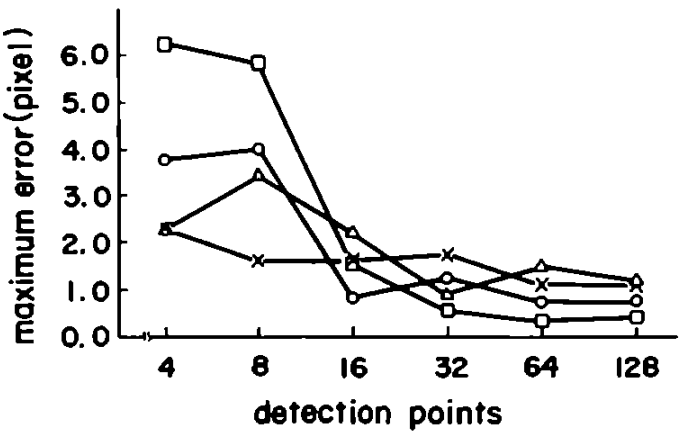
関数として，輝度信号Y を用いた

$$P_t(x,y) = \sum_i |Y_t(x_i+x, y_i+y) - Y_{t-1}(x_i, y_i)| \qquad (2.42)$$

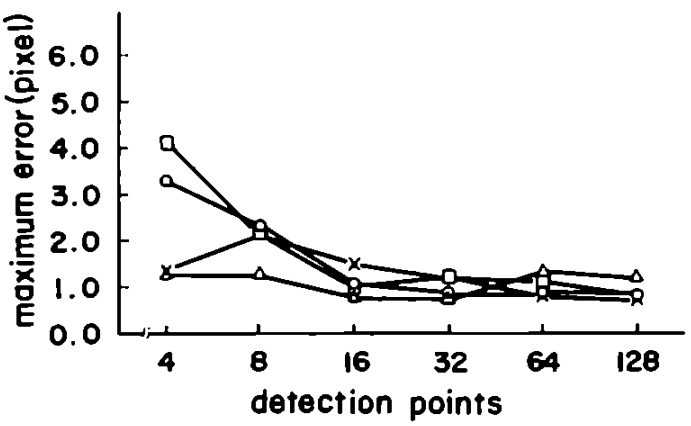
でも動ベクトル検出を行ったが，図２．１７に示したように，（２．41）式の場合に比べ大きな差は見られなかった．従って，（２．42）式を用いた方が演算量の点で有利である．実験の結果，検出点をＹ信号において動物体上の輪郭線近辺のできる限り離れた４エリア（各エリア１５画素以上）に設定すれば，撮影画像においても，±２画素未満の誤差で検出でき，合成処理を行った場合にも問題ないことが明らかになった．

（２）拡大・縮小，回転を伴う場合

（１）と同様に，動ベクトルが既知の動画像（２０フレームの白黒画像）



（ａ）拡大の場合



（ｂ）回転の場合

図２．１８　検出エリアの数と検出誤差

を作成して実験を行った　　．検出点は，動物体内にできるだけ大きい四角形を指定して，この４角形の２本の対角線上に等間隔に配置した．これは，全ての検出点を手動で与えるとすると作業量が多いので，できるだけ自動的に行う部分を増やすための方策である．各検出点（エリア）当り，近傍１６×１６画素を用いて動ベクトル検出を行った．実験結果の１例を図２．１８に示す．図２．１８は，検出エリアの数をパラメータとして，初期フレームで設定した四角形の各頂点が各フレームで対応する位置を，求めた変換行列で算出した値（実験値）と真の位置（理論値）とで比較したものである．最大誤差とは実験値と理論値との距離の２０フレーム中の最大値である．各種画像に対して，回転・ズーム量，検出エリアの位置，検出エリアの数など各種のパラメータを変えて行った実験の結果，対象物上のできる限り大きな四角形を設定し，この四角形の対角線上に３２点以上の検出点を設定すれば，±２画素未満の誤差に収まることがわかった．

次に，実際に撮影して得た動画像に対し，映像合成実験を行った．実験は，カメラをズームしてポスタを撮影した画像，メトロノームの棒の先に小物体

（ａ）撮影画像＋検出点

（ｂ）合成画像

図２．１９　映像合成実験（拡大・縮小，回転を伴う場合）

を貼り付けて動かし，これを静止したカメラで撮影した画像などで行った．図２．１９が，実験に用いた画像の１例である．図２．１９(a)は，ポスタの前に置かれたメトロノームの先に取り付けた小物体が動いている画像で，検出エリアを重ねて表示したものである．(b)は別の背景と合成した２０フィールド後の画像である．手動で全てのフレームについて切り出した場合に比べ±１画素程度のずれがあるが，動きぼけにより，さほど気にならない．撮影画像に対しても，作成画像と同程度の誤差で動物体を抽出できることが明らかになった．

２．４．５　３次元運動する剛体への適用法

本節では，これまでの条件を拡張し，

・車などのように物体自体は形を変えない剛体で，
・滑らかに（突然向きを変えたりしない）移動し，
・物体あるいはカメラの３次元運動によって変形して見える動きで，
・フレーム間であまり激しく形を変えない

場合（これらの仮定は，特殊なものではなく多くの動画像に当てはまる）を対象とし，

『対象領域を２次元アフィン変換が成り立つと仮定できる小領域に分割し，各領域ごとに変換行列を算出して動きを求める』

という方法を検討する．

回転・ズームを伴う場合では，多数の検出点（エリア）で動ベクトルを求め最小二乗法を用いることによって，検出誤差の影響を減らすことができたが，対象領域をいくつかの小領域に分割すると，多数の検出点を設定することが困難になり，十分な検出精度を保つことができない．さらに，複数のフレームにわたる場合は，注目点の各フレームでの正確な対応点が求められないと，フレーム間では小さな誤差でも蓄積されて，最終的には真の対応点からはずれてしまうことがあり，うまく行かない．

そこで，初期フレームにおいて注目点（始点）を手動で設定していたこれまでの補助情報に加え，注目点（始点）の最終フレーム上の位置（終点）も与えることにした．これは，カットの最初と最後のフレームのみ与えればよいので，さほど妥当性を欠いてはいない．

注目点の動きは，図２．２０に示すように始点から終点までの様々な軌跡で表すことができる．この軌跡の中から，最適な軌跡を求めればよい．最適な軌跡を求める問題は，図２．２１の示す多段ネットワークを用いて表現することができる．節点は画面上の位置を表し，枝は動きを示す．始点から終点までの経路が，軌跡となる．最適な軌跡を求めることは，ネットワークの

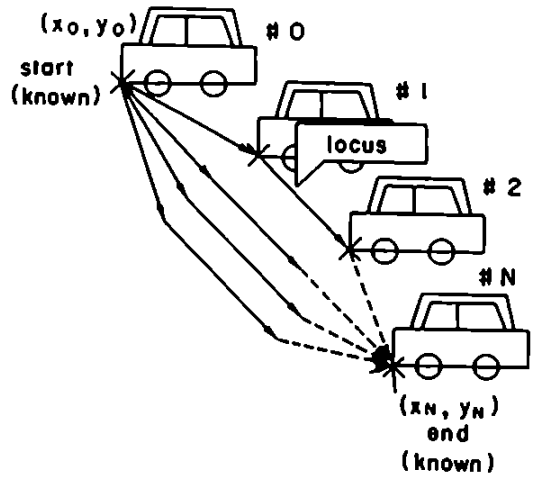


図２．２０　軌跡

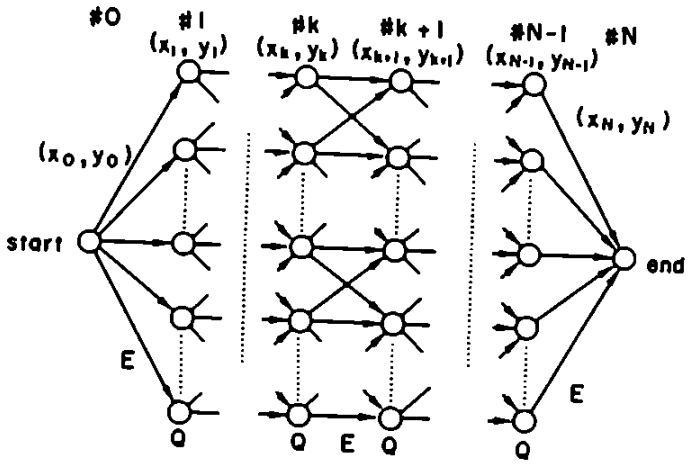


図２．２１　多段ネットワーク

最短路を求めることと等価である．この問題は，動的計画法[36]を用いると，高速に解くことができる．

最短路を求める評価関数として，初期フレームとの近似度と最終フレームとの近似度を表すの評価関数Q_i（各節点の重みに相当）と，物体の動きの滑らかさを表す評価関数E_i（各枝の重みに相当）を使用する．Q_iは，図２．２２に示すように，

$$Q_k(x_k, y_k) = (1-v)\cdot P1_k + v\cdot P2_k \qquad (v=k/N)$$

$$P1_k = \Sigma\Sigma\,|S_0(x_0+j, y_0+i) - S_k(x_k+j, y_k+i)|$$

$$P2_k = \Sigma\Sigma\,|S_k(x_k+j, y_k+i) - S_N(x_N+j, y_N+i)| \qquad (2.43)$$

を用いた．$P1_k$ は初期フレームと現フレームの近似度を，$P2_k$ は現フレームと最終フレームの近似度を，$Sk(x,y)$ は第k フレームにおける位置(x,y) の映像信号レベル，$\Sigma\Sigma$は注目点近傍の全画素に対する総和を，N はフレーム間の数（全フレーム数は N+1である）を示す．Q_kは，初期フレームと最終フレームに類似しているほど，小さな値となる．E_kは，図２．２３に示すように，

$$E_k(\alpha, d) = 1 - \exp\left[-\frac{1}{2}\left(\frac{\alpha^2}{\theta^2} + \frac{d^2}{L^2}\right)\right] \qquad (2.44)$$

を用いた．αは直進を０とした回転角，ｄは移動量，θは回転角の重み係数，

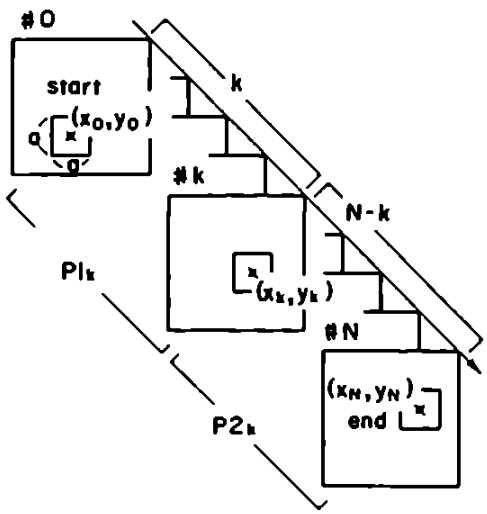


図２．２２　評価関数Q_k

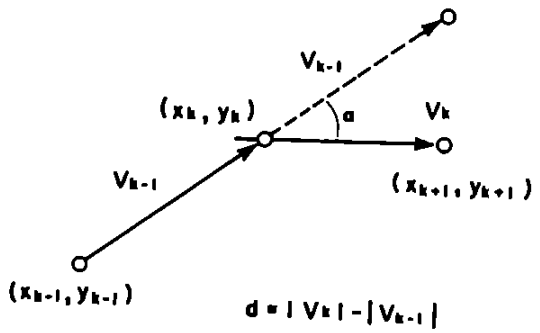


図２．２３　評価関数E_k

L は移動量の重み係数である．E は，動き量が少なく直進するほど小さな値になる．

この評価関数Q と，重み係数 ε を掛けたE を加算した目標関数

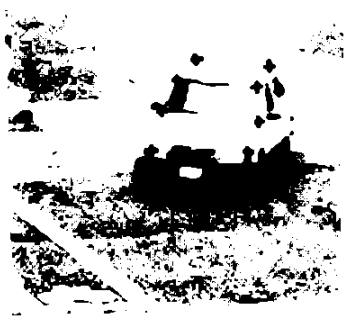
$$\sum_{k=0}^{N-1}\left[Q_{k+1}(X_{k+1},y_{k+1})+\varepsilon\cdot E_{k+1}(\alpha,d)\right] \qquad (2.45)$$

が最小になるパスを求めると，これが最適軌跡である．

本検出法の妥当性を評価するために，本制約条件に適合する撮影画像（レールを走るおもちゃの汽車を撮影した映像，交差点をまがる車をビルの屋上から撮影した映像）に対して動き検出実験を行った[7][8]．実験結果の1例を図2．24に示す．図2．24(a)は初期フレーム，(b)は最終フレームのそれぞれで対応する注目点（＋印）を指定したものある．(c)は，本稿で提案した方法を適用した結果で，(d)は，従来のマッチング法を適用した結果である．実験では，ε は点の重みQの最大値と最小値の差，$\theta=60^{\circ}$，L=6[画素]とした．従来のマッチング法では，注目点が指定した場所から外れ

（a）注目点を指定した
初期(第1)フレーム

（b）注目点を指定した
最終(第12)フレーム



（c）提案した方法
(第7フレーム)

（d）従来の方法
(第7フレーム)

図2．24　検出結果

てしまう場合が多いのに対し，本手法では，注目点が指定した場所から大きく外れることなく追跡できた．本手法の誤差は，筆者が目視で検出したものに比べ，全てのフレームにおいて約２画素の誤差に収まった．

[３]映像信号処理システム のアーキテクチャ

３．１　基本概念

映像信号（動画像）処理の特徴は，膨大な情報量の画像を扱うことにある．例えば，ＮＴＳＣ画像の場合，毎秒数１０Ｍｂｙｔｅものデータを処理しなければならない．これを，汎用のコンピュータで処理することは困難である．その理由は，ノイマン型を採用しているために，①制御が逐次的である，②ＣＰＵとメモリのデータ転送幅が狭いなど，大量の画像データを扱うのに適した構造をしていないためである．この点に関しては，高速に並列演算が行えるスーパーコンピュータにしても同じである．しかし，映像信号処理では，全画面に同じ処理を施す場合が多い，近傍のデータを参照する場合が多いなど一般のデータ処理と異なる特徴があり，映像信号処理の特質を生かしたシステム構成とすれば，高速化が可能となる．本章では，映像信号処理のためのシステム構成について検討を行う．

システム構成法を検討するに当り，画像・映像信号処理に必要な機能について考察する．

（１）幅広い応用分野

放送，通信，工業，医学用など広範な動画像処理，映像信号処理に適用できるようにする．これらの応用では，次に示すような処理がしばしば行われている．

・点演算（濃度値変換，閾値処理など）
・近傍演算（空間フィルタ，輪郭抽出など）
・幾何学変換（拡大縮小，回転，テクスチャマッピングなど）
・色処理（色補正，カラーマトリックスなど）
・フレーム間処理（動ベクトル検出など）
・画像間処理（画像合成，パタン・マッチングなど）

（２）ビデオレート処理

２．１で述べたように，ＮＴＳＣに対しては，１４．３ＭＨｚあるいは，１３．５ＭＨｚ，ＨＤＴＶに対しては，７４．２５ＭＨｚの速度の画素を漏れなく処理することが要求される．ビデオレート処理とは，１フレームの処理が１フレーム期間（１／３０秒）内で終了することであり，さらに次々と入力される画像に対して途切れなく処理を施して出力するものであり，処理時間の関係から入力や出力の流れを止めるといったことは許されない．

（３）複数系統のカラー動画像の処理

放送用では，複数系統のカラー動画像を同時に処理することが求められて

いる．信号形式は，コンポジット信号（ＮＴＳＣ，ＰＡＬ，ＳＥＣＡＭ）あるいは，コンポーネント信号（{R,G,B}，{Y,R-Y,B-Y}，{Y,I,Q}）である．

（４）１６ビット処理

データ幅８～１０ビットの入力画像に対して，処理ステップ数を重ねた場合にも十分な演算精度を保つ必要がある．一方，高速処理のためには，少ない演算語長の方が好ましい．ハードウェア規模も考慮して，通常１６ビットで処理している．

（５）プログラム制御

システムの大半の機能がプログラムで制御される方が好ましい．これにより，従来の専用映像機器と異なり，同一のハードウェアで日々変わる番組制作に対して機能を変更したり，新しい効果を作り出すことができる．さらに，これらのことが，工場の開発・設計技術者によってではなく，番組制作スタジオの技術者によって，現場で行うことができる．

（６）拡張性・運用性

システムの拡張が容易で，新たな機能要求にも柔軟に対応できる構成であること．簡単な処理には，小規模なシステム構成が可能で，複雑な処理には新規にハードウェアを開発するのではなく，小規模システムと同じ基板を使用し，ただ数を増やすだけで対応できることが望ましい．これにより，予備基板の種類が少なくて済み，保守，運用性の向上がはかれる．

３．２　並列処理方式

画像処理は，２次元状のデータ構造や，そのデータ量の膨大さのために，並列処理技術を用いた画像処理専用のアーキテクチャが提案されている．大別すると，パイプライン方式，完全並列方式，局所並列方式，マルチプロセッサ方式になる[42]．

（１）パイプライン方式

図３．１に示すように，同じ時間で処理が完了する異なる機能を有する演算器を多段に接続し，各演算器で処理を行いながら，順に前から後へパイプラインのように送って処理を重ねる方式．この方式を採用しているものに，Cytocomputer[43]，鼻駄天[44]，ＴＩＰ[45]，ＦＩＰ[46]などがある．

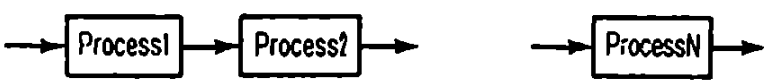


図３．１　パイプライン方式

（２）完全並列方式

図３．２に示すように同じ演算要素（多くの場合２進演算器）を画像の構造に合わせて多数配置して全画素を同時に処理を行う方式．この方式をとるものに，ＭＰＰ[47]，ＣＬＩＰ[48]，ＡＡＰ[49]などがある．

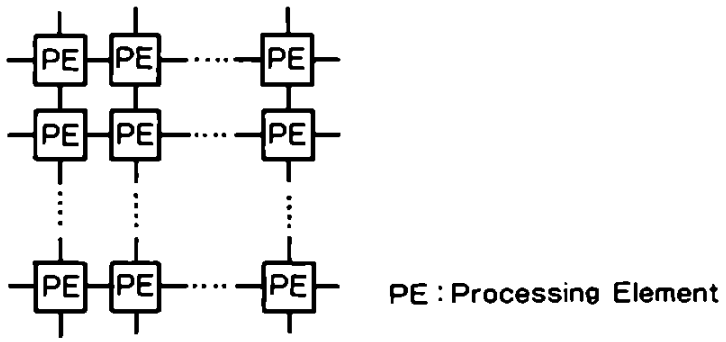


図３．２　完全並列方式

（３）局所並列方式

図３．３に示すように，局所的アレイ（例えば３×３）を全画面あるいは処理対象領域に走査して処理する方式．この方式を採用するものには，ＴＯＳＰＩＸ[50]，ＩＳＰ[51]，ＲＩＳＰ[52]などがある．

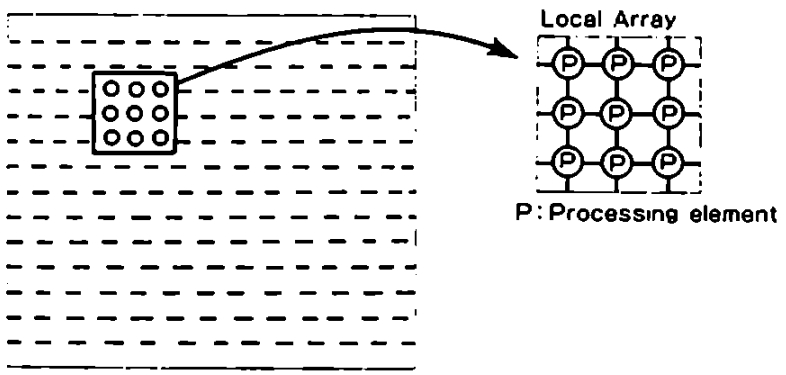


図３．３　局所並列方式

（４）マルチプロセッサ方式

各々のプロセッサが各々のプログラムにより動作する方式．多くの場合，完全独立には動作させず，同期をとりながら処理させる．この方式をとるものに，ＭＡＣＳＹＭ[53]，ＺＭＯＢ[54]，ＳＩＰＳ[55]，ＲＡＰＩＤ[56]，ＶＳＰ[57]などがある．

（１）は，固定のパイプライン構造のため柔軟性に欠ける欠点がある．尊

駄夫[44]では，処理モジュールの接続を変えられるネットワークを採用し，パイプライン構造を可変にできるようにしているが，処理形態はパイプラインのままであり，用意した処理モジュールの機能以外は実現できない．負荷の大きいまとまった処理を複数のプロセッサに分散して行うような用途には向いていない．また，（２）は，画素数に相当する多くのプロセッサ数が必要となり，かつプロセッサ間の接続が固定的で離れたプロセッサ間の通信に時間がかかり，制御もＳＩＭＤ（Single-Instruction Multiple-Data）のために柔軟性に欠ける．（３）は，画像処理で多用される近傍画素演算（空間積和演算）に適しており，小規模のハードウェアで構成できるため，商用の画像処理装置に多くみられる方式であるが，幾何学変換のように近傍画素以外のデータを扱う大局的な処理には，向いていない．

３．１で述べた機能を実現するためには，制御方式は多少複雑となるが，処理に応じて接続および処理形態を自由に変えられる柔軟性に優れたマルチプロセッサ方式の方が望ましい．

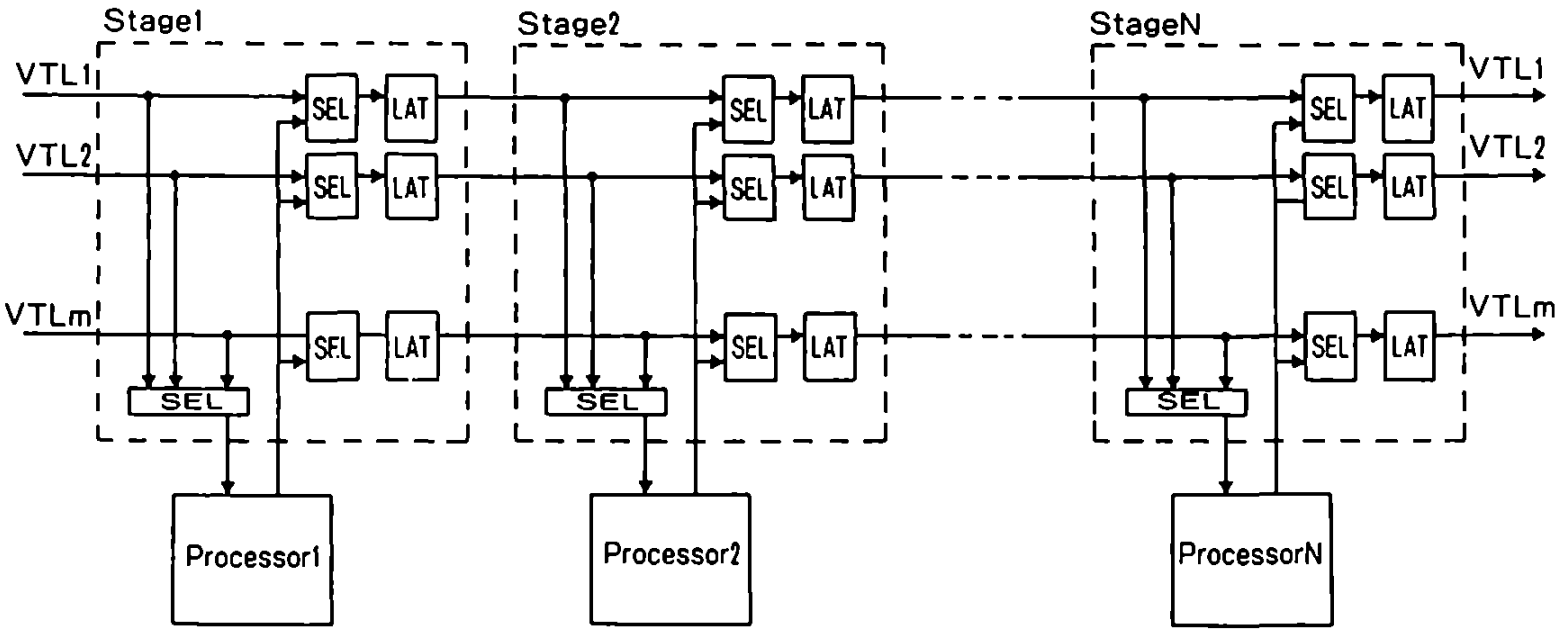
一口に，マルチプロセッサ方式といっても，プロセッサ間の結合形態の違いから，様々な方式がある[56][59]．１つの画像を細かなブロックに分割しブロック毎にプロセッサを割り当て，これを並列に動作させることがＶＳＰ[57]などでは，行われている．この場合は，直並列変換を行うことにより，少ない本数のバスでプロセッサ間接続が実現できる．しかし，複数の映像信号間や色信号間の処理を行う場合には，複数の映像信号を通すための多数のバスが必要である．また，処理によっては，処理途中の中間映像のためにバスが必要である．一方，多くのフレームに渡る処理では，プロセッサを多段に接続する必要がある．バスの本数が，プロセッサの処理能力以上に，システムの処理能力を制約しかねない．このため，プロセッサ間の接続は，自由であることが望まれる．しかし，プロセッサ間をすべて結ぶとなると，プロセッサ数ｎに対して，接続数Ｌは，

$$L = {}_nC_2 = n(n-2)/2 \qquad (3.1)$$

のようにn²のオーダーで増大し，あまりに多くの線が必要になり，製作上問題である．

そこで，次に示す２つの構成法を考案した．

ConfigurationA　図３．４に示すように，プロセッサは，複数の単方向性の映像転送路によって結合される．映像転送路上では，映像データは，走査された映像信号列となっている．各プロセッサは，映像転送路から必要なデータが自分のプロセッサに流れて来たタイミングに信号列から選択して入力し処理する．処理結果は，映像信号列となるタイミングに映像転送路に出

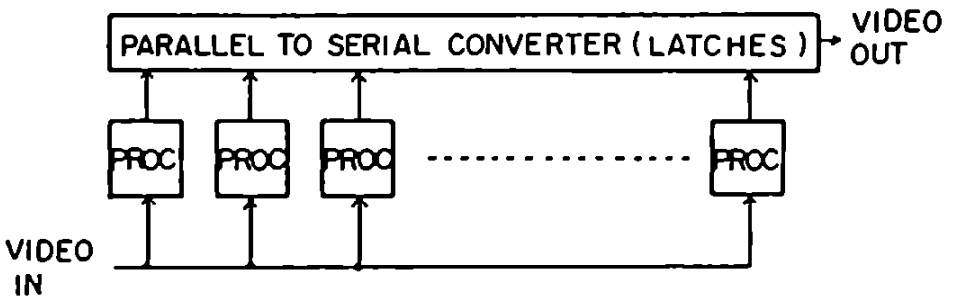


VTL : Video Transfer Line, SEL : Selector, LAT : Latch
（映像転送路）

図３．４　構成法 ConfigurationA

力する．手前のプロセッサから来たデータをそのまま通過させ，次のプロセッサで処理させることもできる．映像転送路には，プロセッサ毎にラッチが挿入されており，映像信号の標本化周波数のクロックで動作している．セレクタとラッチの動作により，図3．5に示すようにパイプライン・モードにも，パラレル・モードにも適応できる．この方法によれば，転送路への出力は映像信号列となるように決められているので，バスアクセス競合もなく，入出力もスムーズに行える．このため，プロセッサの台数の増大に比例した処理能力の向上を望める．また，映像転送路は，ラッチによって規制されているため，通常のバス方式で生じる線路長の増大によるプロパゲーション・ディレイの問題がない．物理的には，プロセッサの接続台数に制限がなく拡張が容易である．さらに，映像転送路では，データは映像信号列となっているために，映像転送路の間にD／A変換器を接続することにより，処理途中の状態をピクチャモニタで直接見ることができる．これにより，操作性，保守性を向上させることができる．この構成法は，4章で述べる実験システムRTVPで採用した．

（a）パイプライン・モード



（b）パラレル・モード

図3．5　動作モード

Configuration B　図3．6に示すように，16個のプロセッサをひとまとまり（クラスタ）にして，この中では完全自由に接続できるようにし，このクラスタを多段に接続される．クラスタ間は，16本の単方向性のバスで接続した．

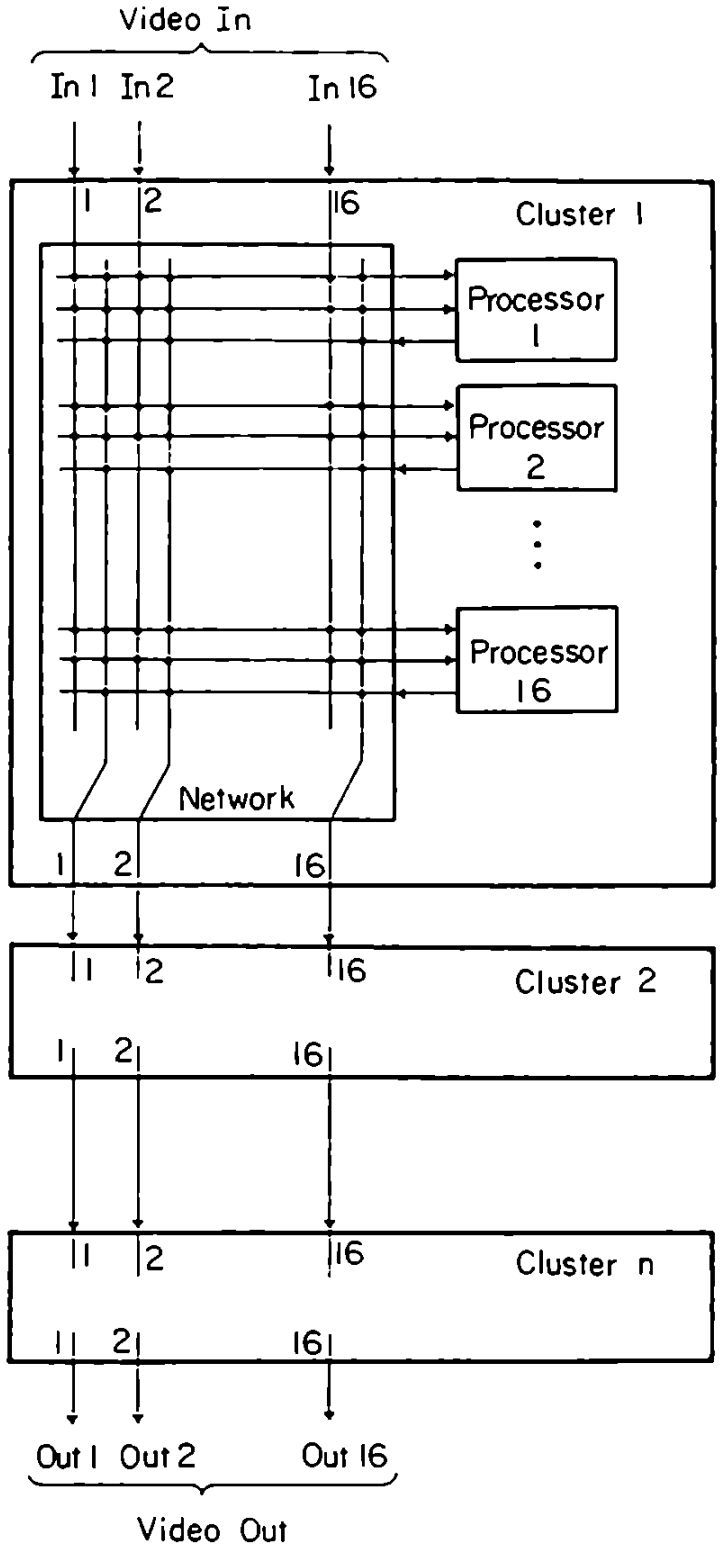
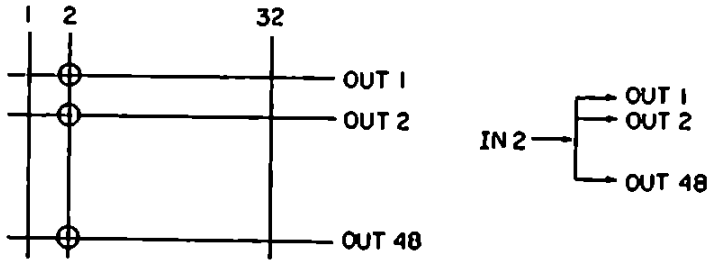
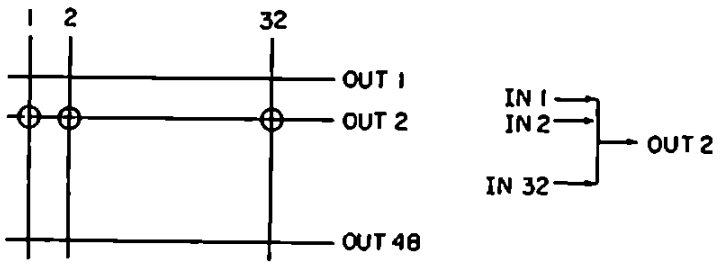


図3. 6 構成法 ConfigurationB

ネットワークは，３２入力（プロセッサからの出力１６とクラスタ外からの入力１６）４８出力（プロセッサへの入力３２とクラスタ外への出力１６）のクロスバー・マトリックスであり，入力と出力の任意の接続が可能である．図３．７に示すように，１つの入力を同時に複数の出力に分配すること（ブロードキャスト・モード）や，複数の入力を時分割に１つの出力に集中すること（コンセントレイト・モード）が可能である．これにより，プロセッサをパイプライン状に接続することや，アレイ状に接続することも，さらに両者を組み合わせた形態をとることもできる．クラスタをまたがって，複数の映像入力を並行して供給したり，処理結果の映像や処理途中の映像も出力でき，処理能力をあまり犠牲にすることなく，パイプライン処理やパラレル処理を実現することができる．また，クラスタ内では，フィードバックループを構成することも可能で，巡回型の処理も容易に行える．この構成法は，５章で述べる実用システムＰｉｃｏｔ－ｓｙｓｔｅｍで採用した．



（ａ）ブロードキャスト・モード



（ｂ）コンセントレイト・モード

図３．７　接続モード

３．３　メモリ構成法

マルチプロセッサシステムにおいては，メモリの配置の方法によって，処理能力は大きく変化する．メモリ配置の方法を大別すると，

（１）共有方式

（２）分散方式

になる.

メモリを1ヶ所に集中させ，これを複数のプロセッサで共有する共有方式では，複数のプロセッサが同時にメモリにアクセスすると，メモリアクセス競合が生じて，処理能力が低下する．このため，メモリを分割し複数バスやクロスバー接続にして，同一メモリに同時にアクセスする確率を下げたり，バスの伝送容量を増やしたり，アービタを設けたりして，競合を緩和する工夫がされている．しかし，画像処理のように，メモリが大容量でかつ頻繁にアクセスされる処理では，必要な接続線は膨大なものになる．クロスバー・スイッチの場合の規模は，データ幅dビット，アドレス幅aビット，プロセッサ数p，メモリ数mとすると，接続線数はプロセッサとメモリを分離して配置した場合，(d+a)×p×mとなり，大規模になる．

これに対して，各プロセッサにローカルメモリをもたせてデータを分散して格納する分散方式は，メモリアクセスによる競合は生じない．しかし，自分のメモリにないデータが必要になった場合，必要なデータをもつプロセッサと通信を行わなければならない．この通信が，あまり頻繁に行われると速度低下を招く．画像を固定的にブロックに分割してメモリ分散させてしまうような方式をとると，幾何学変換に際し，プロセッサ間で通信が頻繁に起こることになり，効率が下がる．

このため，分散方式を基本とするが，プロセッサが処理に必要とするデータは，各プロセッサのメモリに全て確保しておく方式とした．必要なデータの配置は，処理の内容によって前もってわかっているので，処理に先立ち自分のメモリに格納しておく．処理のやり方によっては，プロセッサが必要とするデータが複数のプロセッサで同じであることが起こりうるが，その場合は，別々のプロセッサに同じ内容のデータが確保されることになる．一見無駄のようであるが，メモリアクセス競合や通信手続きが不要のため，プロセッサ数に比例した処理能力向上が望める．メモリの容量は，幾何学変換を行うのに最低必要な1フレームあるいは1フィールド（インタレース走査している映像信号では，1フレームの半分の容量）とすることにした．

この考え方は，離散系シミュレータKDDS－1[60]のマルチリードメモリや行列計算用のシステム[61]のブロードキャストメモリに似ている．書き込みに比べ読み出しの頻度が多い場合に有効であり，前述のように近傍演算の多い映像信号処理の特質に類似している．しかし，マルチリードメモリ方式やブロードキャストメモリ方式と異なり，メモリの書き込みに際してその都度コピー動作を行うことはしない．処理は出力画像を基本に行うために，自分と同じ出力対象領域を処理しているプロセッサは他にない．このため，処理結果をほかのプロセッサに伝えなくても，同一のフレームあるいはフィ

ールド内の処理に限れば問題は生じない．プロセッサの処理がフレームあるいはフィールド単位で完了した後，処理を行ったプロセッサから書き換えられた部分のデータを集めて，矛盾の無い完成したデータとする．

このメモリ構成法は，実験システムＲＴＶＰ，実用システムＰｉｃｏｔ－ｓｙｓｔｅｍの両者で採用した．

３．４　同期方式

多数のプロセッサを使用するマルチプロセッサシステムにおいては，処理過程でさまざまな遅れが生ずるので，プロセッサ間の同期の問題について考慮する必要がある．例えば複数のプロセッサを図３．８に示す系統で接続して処理を行った場合，以下の解決すべき問題が生ずる．

Problem１　パスＡ，Ｂ，Ｃ間で位相にずれがあるため，これを補正する必要がある．

Problem２　同じシステム同期信号で全てのプロセッサをプログラム制御すると，多数のプロセッサを通った場合，映像のみが遅れ同期と合致しなくなる．このため，画面の表示部分が欠落し，有効画素数が不足することがある．

Problem１ のためには，各プロセッサに遅延量補償要素を設ければ良い．後述のＰｉｃｏｔでは，各プロセッサに最大１ラインの容量を持つ可変遅延をもたせている．**Problem２** のためには，次の２つの対策を講じた．

MeasureＡ　処理に応じて後段のプロセッサほど遅れたシステム同期信号を供給する．システム同期信号は，処理に応じてプロセッサ毎に変えるのが望ましい．しかし，これはハードウェア製作上も制御上も大きな困難を生じるので，個々のプロセッサではなく，ある程度まとまった数のプロセッサ（後述のＰｉｃｏｔ－ｓｙｓｔｅｍでは，１６台）で共通にする．このため，

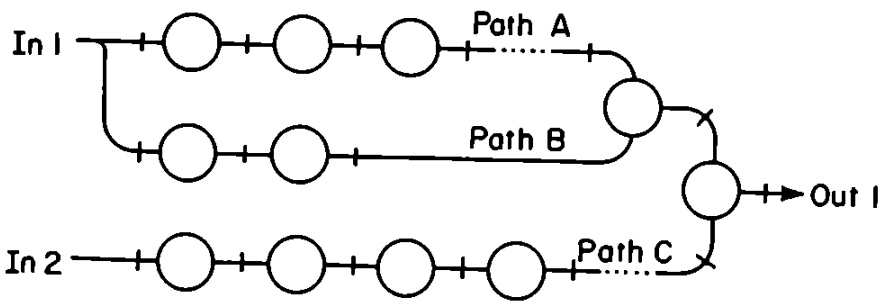


図３．８　パスの違い

プロセッサによっては，映像とシステム同期が完全には一致しない場合が生じ，有効画面が欠落することがある．これを避けるために，水平同期信号の幅を本来の同期信号規格よりも狭くする．

Measure B　処理される映像に同期情報を付けて，この同期情報を使ってプロセッサを制御する．この同期情報は，プロセッサのプログラム制御回路に入力し，これを用いてプロセッサを動作させることにより，映像と同期を合致させた処理を行うようにする．この対策は，ＲＴＶＰ，Ｐｉｃｏｔ－ｓｙｓｔｅｍ共に採用しているが，具体的な構成法については４章，５章で述べる．

３．５　プログラムロード方式

放送においては，演出効果を上げるために「色変換を行った後に幾何学変換を行う」といったように，次々と処理内容を変えて行くのが普通である．これを行っても画面上に切り替えノイズ等の妨害を生じてはならない．この要求を満たすためには，映像信号の垂直帰線期間に切り替えを行うようにしなければならない．しかし，垂直帰線期間は，約１ｍ秒と短いために，長いプログラムのロードは困難である．このため，ＲＴＶＰでは，プログラムメモリ全てをダブルバッファ化し，切り替えは垂直帰線期間に行うが，プログラムのロードは垂直帰線期間以外でも可能とした．変更されたプログラムは，変更の直後に来た垂直帰線期間に実行を開始する．Ｐｉｃｏｔ－ｓｙｓｔｅｍでは，プログラムメモリはダブルバッファとしないで前もって異なる処理を行うプログラムモジュールを多数プログラムメモリにロードしておき，実行したい処理のプログラムの先頭番地と必要なパラメータのみを与えるようにした．この先頭番地とパラメータを受け渡すためのメモリは，ダブルバッファ化し，いつでもロードできるようにした．

[４]実験システムＲＴＶＰ

４．１　設計指針

ＲＴＶＰ（Real-time video signal processor）は，複数の同一仕様のプロセッサを並列動作させることで映像信号をビデオレートで処理することが可能であるかを検証するための実験システムとして設計した．従って，処理の対象画像も放送品質ではなく３．５８ＭＨｚと低レートでかつ２入力の白黒映像と小規模のものとした．また，プロセッサ台数も，並列処理やソフトウェアによる機能変更の容易性など基本的な事柄を検証するに足るプロセッサ数８台とした．従って，放送で使われる実用的な処理を全てについてはチェックはしないこととした．このため，将来のＬＳＩ化について配慮した回路構成とするが，ＬＳＩの開発は行わず市販の個別部品を使用して製作した．

４．２　アーキテクチャ

４．２．１　概要

ＲＴＶＰは，図４．１に示すように，３本の映像転送路（ＶＴＬ，Video transfer Line）により多段に結合されたプロセシングユニット群とホストコンピュータ（ミニコンピュータＭＶ４０００）から構成される．映像転送路は，それぞれ１６ｂｉｔ幅である．２系統の映像信号入力は，それぞれＡ／Ｄ変換部によって標本化周波数３．５８ＭＨｚ，量子化数８ｂｉｔにディジタル化され，３本の映像転送路のうちの２本に供給される．プロセシングユニットは，映像転送路から映像データを取り込んで処理し，処理結果を再び映像転送路に出力する．プロセッシングユニット群で処理された映像データは，映像転送路の終端でマルチプレクサによって１つが選択され，Ｄ／Ａ変換部によってアナログ映像信号に変換される．

映像転送路では，映像信号の標本化周期（２８０ｎｓ）で，データが図４．１中の左から右へ流れていく．一方，プロセシングユニットは，１４０ｎｓのクロックサイクルで動作している．従って，各プロセシングユニットは，１画素当り２ステップに相当するプログラムの処理を施して映像転送路に出力する．

ホストコンピュータは，各プロセシングユニットのマイクロプログラムの作成およびプログラムメモリへのロード，ルックアップテーブルのデータ作成および各プロセッシングユニットのルックアップテーブルメモリへのロード，映像転送路の終端にあるマルチプレクサなどマイクロプログラム以外のパラメータの制御，マイクロプログラムのデバック用に用意したシングルステップ動作などの制御を行う．

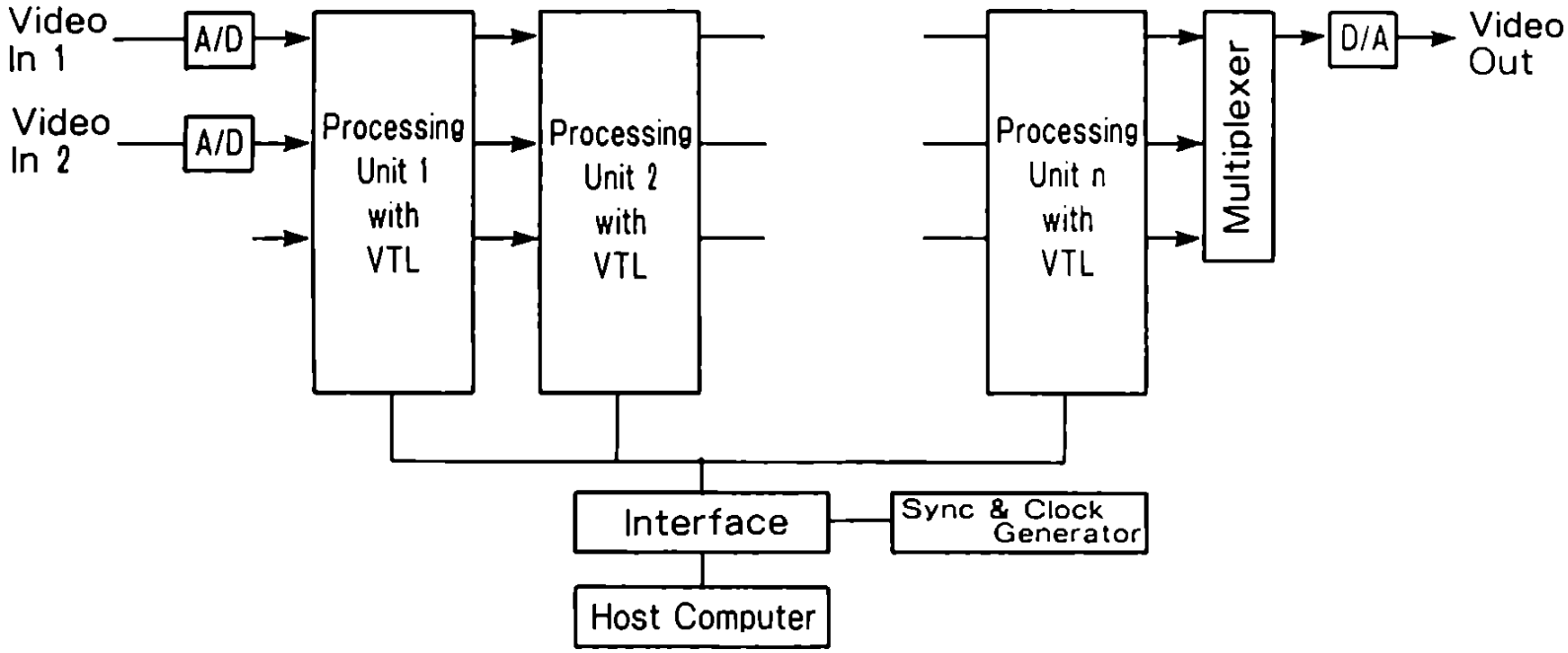


図4．1　ＲＴＶＰの構成図

表４．１に主な仕様を，図４．２にプロセシングユニット８台構成のシステムの写真を示す．

表４．１　ＲＴＶＰの仕様

映像入力	標準ＴＶ（白黒信号）　２系統　各８ｂｉｔ
映像出力	標準ＴＶ（白黒信号）　１系統　１０ｂｉｔ
標本化周波数	３．５８ＭＨｚ
データ語長	１６ｂｉｔ
処理速度	１４０ｎｓ／ｐｉｘｅｌ，３３ｍｓ／ｆｒａｍｅ
映像転送路	３本　各１６ｂｉｔ （映像信号１２ｂｉｔ＋同期信号４ｂｉｔ）
プロセシングユニット数	８台
プロセシングユニット仕様	ＡＬＵ（１６ｂｉｔ） 乗算器（１６×１６→３２ｂｉｔ） ルックアップテーブル（2^{12}×１２ｂｉｔ） 画像メモリ（１２８ｋ×１２ｂｉｔ） マイクロプログラム制御（２５６ｗ×２７２ｂｉｔ） クロックサイクル　１４０ｎｓ

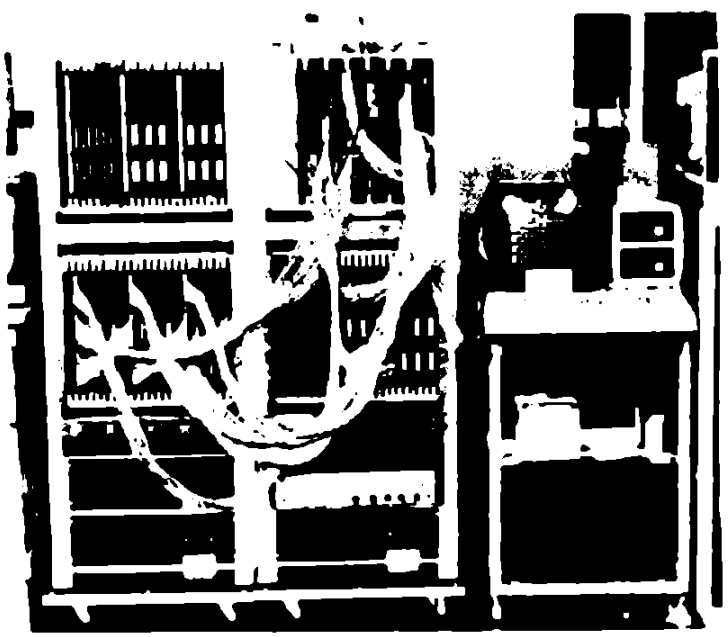


図４．２　ＲＴＶＰの外観写真

4．2．2　映像転送路

ＲＴＶＰでは，３．２で提案したConfigurationA を採用した．映像転送路は，映像入力より１本多い３本とした．映像転送路の転送サイクルは２８０ｎｓで，データ幅は１６ｂｉｔである．このうち，映像情報は１２ｂｉｔで，残り４ｂｉｔは３．４のMeasureB で述べた同期情報（水平同期ＨＤ，垂直同期ＶＤ，フレームパルスＦＰ，フラッグＦＬＡＧ）に割り当てた．映像転送路のセレクタとラッチは，プロセシングユニットに含めた．これにより，プロセシングユニットの増設のみでシステム規模の拡大が可能である．また，プロセシングユニット間にＤ／Ａ変換器を直接接続することにより，各ユニットの出力を見ることができ，処理途中の状態をＴＶモニタで容易に確認できる．

4．2．3　画像メモリ構成法

画像メモリは，３．３に述べたように，各プロセシングユニットに持たせた．プロセシングユニット当りのメモリ容量は１２８ｋ×１２ｂｉｔである．本システムでは，映像信号を３．５８ＭＨｚで標本化しているために，１フレームに相当する容量である．画像メモリは，１４０ｎｓのサイクルタイムでランダムアクセスが可能である．映像信号の標本化周期，映像転送路の転送サイクルは，２８０ｎｓの周期なので，１サイクルにつき２回のメモリアクセスが可能である．通常は，前半をプロセシングユニット側の読み出しサイクル，後半を映像信号転送路からの書き込みサイクルとしている．書き込みと読み出しは別々のアドレス発生回路が行う．映像転送路内では，データは走査線構造をしていることから，その書き込みアドレス発生回路はカウンタ回路だけでよい．映像信号とともにアドレスカウンタのクリアのための水平同期パルスと垂直同期パルスを転送し，インタフェースの簡素化をはかる．この機構は，次節で述べる時間軸補正処理にも役立っている．

4．2．4　２段階時間軸処理方式

放送局内の映像信号と局外の中継車などから送られてくる映像信号を合成する場合，同一の同期信号で駆動されていないために，直接合成処理することはできない．このような非同期の映像信号を同時に扱うために時間軸補正（フレームシンクロナイズ）機能を実現した[4]．

入力映像信号の標本化周期を$t_{s_{in}}$，水平同期の周期を$t_{H_{in}}$，垂直同期の周期を$t_{V_{in}}$，システムを駆動している標本化周波数を$t_{s_{out}}$，水平同期信号を$t_{H_{out}}$，垂直同期信号を$t_{V_{out}}$とすると，入力信号を基準同期に変換するためには，

$$t_{s_{in}} \rightarrow t_{s_{out}},\ t_{H_{in}} \rightarrow t_{H_{out}},\ t_{V_{in}} \rightarrow t_{V_{out}} \qquad (4.1)$$

のように変換しなければならない．これらの値には次の関係がある．

$$t_{vw} = 525 \times t_{hw} \quad (4.\ 2)$$

$$t_{hw} = n \times t_{sw} = (n1+n2)\, t_{sw} \quad (4.\ 3)$$

$$t_{vr} = 525 \times t_{hr} \quad (4.\ 4)$$

$$t_{hr} = n \times t_{sr} = (n1+n2)\, t_{sr} \quad (4.\ 5)$$

ここで，nは1ライン中の画素数であり，n1は水平有効画素数，n2は水平帰線期間中の画素数を示す．3．58MHzの標本化の場合には，n＝227.5，n1＝189，n2＝38.5である．

従来のFSやTBCでは，この変換をすべて1つのバッファメモリで実現しているが，その制御回路はかなり複雑である．それは，位相や周期の異なる2系統のクロックを同時に扱わなければならないためである．これを簡略化するために，

STEP1　$t_{sw} \rightarrow t_{sr}$　(4. 6)

STEP2　$t_{hw} \rightarrow t_{hr}$，$t_{vw} \rightarrow t_{vr}$　(4. 7)

のように，標本化周波数だけをまず変換し，次に水平と垂直の各周期を変換するという2段階方式で時間軸処理する．中間段階（STEP1とSTEP2の間）では，t_{hw}とt_{vw}がそのままで，t_{sw}だけt_{sr}に変更されるため，（4．5）式の関係が成立しなくなる．そこで，

$$t_{hw} = (n1+n2')\, t_{sr} \quad (4.\ 8)$$

のように水平帰線期間の画素数n2'を削ったり増やしたりして調整する．これにより，A／D変換部では，STEP1のクロックの調相を行うので済み，大きなメモリは不要となり，映像転送路も含めたプロセシングユニットは，同一のクロックで駆動することが可能となる．STEP2に関しては，4．2．3で述べたように映像転送路の同期パルスによってメモリへの書き込みを行い，読み出しをシステム同期で行うことにより自動的にSTEP2が実現される．

4．2．5　プログラムロード方式

3．5で述べたように，画面上に妨害を与えないで処理内容を変更するために，プログラムの切り替えは垂直帰線期間に行う．RTVPでは，後述のようにプログラムは，1プロセッシングユニット当り256ワード（各272bit）のマイクロプログラムである．垂直帰線期間（約1ms）内に8台のプロセシングユニットにロードするためには，約70MByte／secの転送レートが要求されるが，この転送レートを実現するのはハードウェアの設計上大きな困難を伴う．

そこで，各プロセッサにバッファメモリを設け，ホストコンピュータからバッファメモリへは垂直帰線期間に限らず，随時プログラムのロードができ

るようにした．ホストコンピュータとバッファメモリ間は１６ｂｉｔ幅のバスで転送されるので遅いが，バッファメモリとプログラムメモリ間は２７２ｂｉｔ幅のバスで転送されるため，短い垂直帰線期間でも容易に転送が完了する．また，バッファメモリはプログラムメモリの４倍の容量を確保したため，あらかじめ機能の異なるプログラムをバッファメモリにロードしておけば，バッファメモリからどの部分をプログラムメモリに転送するのか指示するだけでもプログラム切り替えが可能になった．さらに，プロセッシングユニットは，並列動作させるためプロセッサのプログラムの内容はほとんど同じでパラメータのみ異なるといった場合が多いので，ホストコンピュータから各プロセシングユニットにブロードキャストモードでロードできるようにした．

４．３　ハードウェア構成

４．３．１　プロセシングユニット

プロセシングユニットＰＵ（Processing Unit）は，図４．３に示すように映像転送路ＶＴＬ（Video Transfer Line）も含んだ構造をしている．ディジタル化された映像信号をＶＴＬより書き込み制御回路ＷＣ（Write Controller）を用いて，画像メモリＩＭ（Image Memory）に書き込む．書き込まれた画像データをアドレス演算部ＸＰ（X address Processing part），ＹＰ（Y address Processing part）を用いて読み出す．読み出されたデータは，データ演算部ＤＰ（Data Processing part）で処理し，結果をＶＴＬに出力する．これらの制御は，ＰＵ内の制御部ＣＴＬ（Controller）が行う．ＶＴＬは，セレクタとラッチで構成されている．試作では３本のＶＴＬを設けた．

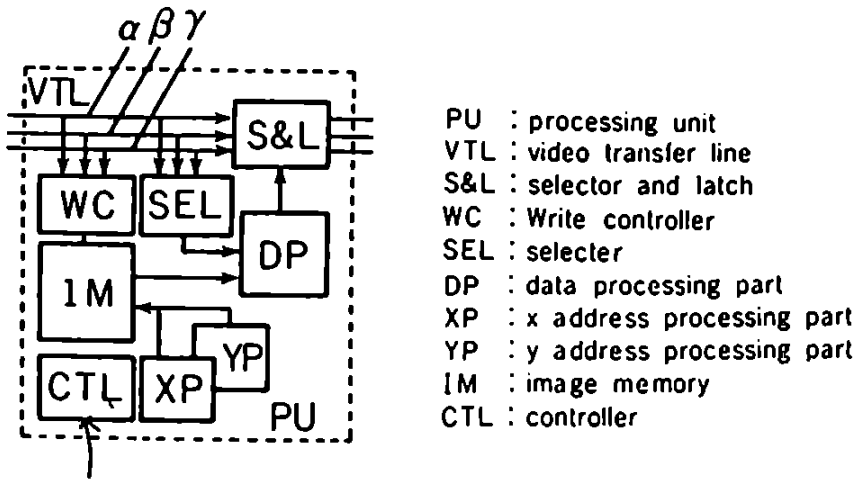


図４．３　プロセッシングユニットの構成図

ＩＭは１フレームの容量をもつ．ＩＭへのデータ書き込みは，ＶＴＬ上を映像信号と共に送られてきた同期信号でＷＣが行う．読み出しは１４０ｎｓのサイクルでランダムアクセスが可能であり，書き込みと独立して，ＸＰ，ＹＰが行う．

ＤＰ．ＸＰ．ＹＰは同一仕様となっており，ＡＬＵ（１６ｂｉｔ），乗算器（１６×１６→３２ｂｉｔ），ルックアップテーブル（４０９６ｗ×１２ｂｉｔ）の演算要素を有する．これらは，複数バスによって結合されており，相互にデータ交換が可能である．単一共通バスに比べ信号線が増える欠点はあるが，演算要素を並行して使用することができるためスループットは良くなる．この複数バスを使うことにより，アドレス演算部ＸＰ，ＹＰはアドレス処理だけするといったように固定的な使い方だけでなく，処理に余裕がある場合は負荷分散させることができる．

プロセッシングユニットは，プロセシングユニット内の制御部ＣＴＬによって動作する．このＣＴＬには，ＲＡＭで構成されたプログラムメモリがあり，このプログラムに従って動作する．このプログラムメモリをホストコンピュータから書き換えることによって，各種の処理を行う．

ＣＴＬは，水平型のマイクロプログラム制御方式を採用している．ＩＰＵにつき２７２ｂｉｔ×２５６ｓｔｅｐの容量である．このマイクロプログラム制御方式はデータ量が大きくなる欠点はあるが，全ての演算要素を同時に制御できる利点がある．プログラムメモリは，ホストコンピュータから見ると１６ビット幅となるように分割され，ルックアップテーブルと共に連続したアドレスをもつメモリ空間に配置した．ＰＵ番号，アドレス，データ（各１６ｂｉｔ）をひとつのパケットとすることにより，変更部分のみをホストコンピュータより転送することが可能となる．また並列動作させるため，各ＰＵのプログラムのほとんどは同じで，パラメータのみ異なるケースが多いので，ブロードキャストモードの転送も可能とした．プログラム変更は，映像信号に切り替えノイズを生じさせないように垂直帰線期間に行うが，前述のように，ホストコンピュータからのプログラム転送は常時行えるように，バッファを設けている．このバッファは４バンクすなわち１ｋ×２７２ｂｉｔあり，この範囲内であればバンク切り替えのみでプログラム変更ができる．

１台のＰＵは，ほぼＢ４版大の基板を８枚（ＤＰ，ＸＰ，ＹＰ各２枚，その他２枚）使用して実現した．

４．４　ソフトウェア構成

４．４．１　処理の流れ

ＲＴＶＰは，３．５８ＭＨｚの標本化レートに対して，７．１６ＭＨｚの

命令サイクルであるので，実時間処理を行うためには，ＩＰＵでは１画素当り２ステップの命令しか実行できない．しかし，ＰＵを並列に動作させることにより，等価的に１画素当りＰＵ数×２ステップのプログラムを実行することができる[4]．ＰＵ群は，アレイ的にもパイプライン的にも動作させることができる．この並列動作を始めとするＲＴＶＰの動作方式について，混合処理とグラディエント処理を例にとって説明する．

（１）混合処理

２つの映像入力ＡとＢの混合処理は，

$$M(i,j)=k\times A(i,j)+(1-k)\times B(i,j) \quad (0\leq k\leq 1) \qquad (4.9)$$

で示すことができる．ここで，A(i,j)は映像Ａのアドレス(i,j)における濃度値，B(i,j)は映像Ｂのアドレス(i,j)における濃度値，M(i,j)は混合結果Ｍのアドレス(i,j)における濃度値，ｋは混合比である．この処理を実現する方法は，いろいろ考えられるが，ここでは図４．４に示すようにタスクを分割して行う場合を考える．すなわち，Ａをｋ倍してとして出力するTASK1，Ｂを(1-k)倍してＤとして出力するTASK2，このＣとＤを加算しＭを出力するTASK3 である．この処理のフローチャートは，図４．５に示すようになる．すなわち，

TASK1　ＩＭには映像Ａを書き込むようにしておく．まず，乗数ｋを乗算器のパラメータｐに置数する（step1）．ＩＭのアドレス値x,y をどちらも１とし，垂直同期ＶＤをチェックする（step2）．垂直帰線期間終了し，かつ水平帰線期間が終了したら（step3），アドレス(x,y) の画像データＩＭ(x,y)を読み出しｋ倍すると共に，ｘを１増やす（step4）．次に乗算して得られた結果ＣをＶＴＬαに出力すると共に，水平同期ＨＤのチェックを行う（step5）．step4とstep5を繰り返し，水平帰線期間になったらx を１として，ｙを１増やす（step6）．つまり，次のラインに移るわけである．水平帰線期間をチェックして，水平帰線期間が終了したら，再びloop1 にはいる（step7,8）．

TASK2　TASK1 と同様の処理を映像Ｂに対して行い，結果Ｄを出力する．ただし，乗数ｐは1-kとする（step1）．

TASK3　ＣとＤを加算する処理で，ＶＴＬαとＶＴＬβからデータを取り込み（step3,4），これらを加算して（step5），結果をＶＴＬγに出力する（step6）．これを水平帰線期間になるまで続ける（loop3）．垂直帰線期間，水平帰線期間のチェックは，TASK1，TASk2と同様に行う（step1,2,7,8）．

ここで，出力のためのループに何ステップ必要なのかによって割り当てるＰＥ数が決まる．TASK1 については，loop1 がstep3 とstep4 のステップに

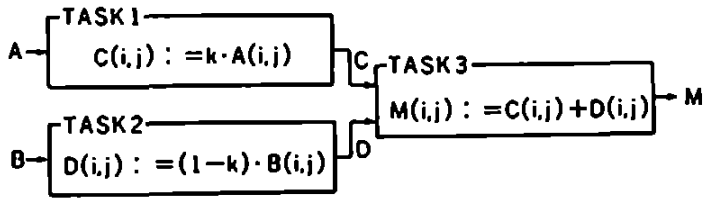


図４．４　２映像の混合処理

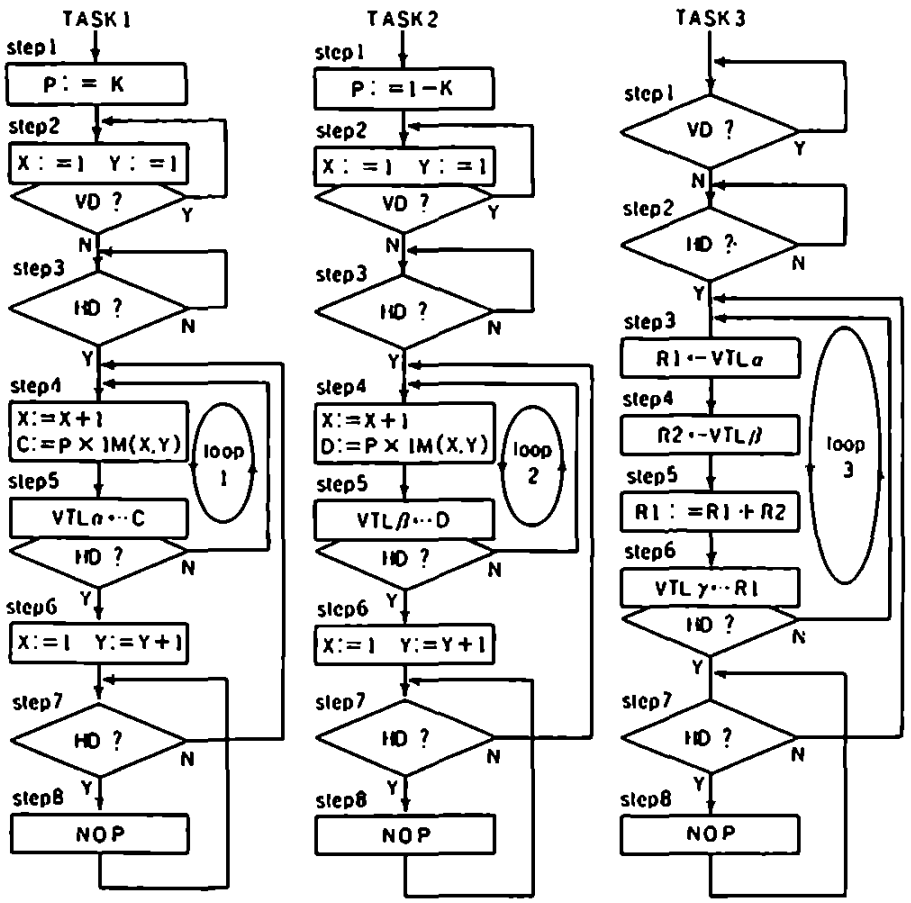


図４．５　混合処理のフローチャート

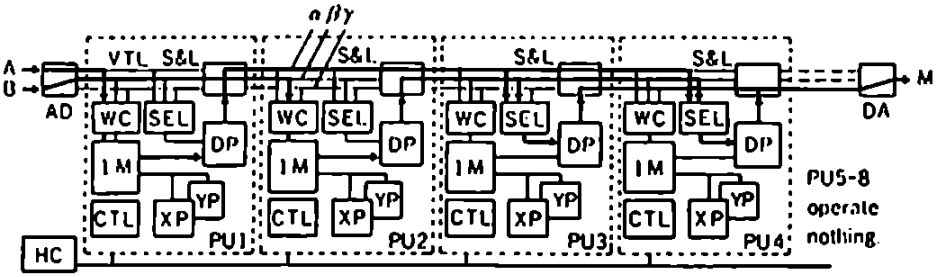


図４．６　混合処理の信号系統

対応しており１画素当り２ステップとなっているので，１個のＰＵで処理できる．TASK2 についても，TASK1 と同様であり，１個のＰＵで処理できる．TASK3 については，loop3 にstep3～6の４ステップ必要であるので，２個のＰＵが必要である．

信号の流れを図４．６に示す．図４．６では，TASK1 にＰＵ１を，TASK2 にＰＵ２を，TASK3 にＰＵ３，ＰＵ４を割り当てている．ＰＵ３と４は，ＶＴＬ.αよりCをＶＴＬ.βよりDを取り込み加算して出力する処理を同期して並列に行う．出力のためのstep6 は，４ステップ毎に繰り返される．４ステップ毎にＰＵ３がM_1,M_5,M_9,・・・を，ＰＵ４がM_2,M_6,M_{10},・・・を順にＶＴＬ.γに出力すると，ＶＴＬ.γ上にはM_1,M_2,M_3,M_4,・・・と順に隙間のない映像信号系列となる．この際，ＰＵ３とＰＵ４の出力は同時に行えばよい．このように，ＰＵ３と４は，同じタスクをアレイ的に処理している．これに対しＰＵ１とＰＵ３，４との関係，及びＰＵ２とＰＵ３，４の関係は，パイプライン的に動作していると言える．

（２）グラディエント

グラディエントのひとつであるRoberts オペレータは，

$$g(i,j)=|f(i+1,j+1)-f(i,j)|+|f(i+1,j)-f(i,j+1)| \qquad (4.10)$$

で示される．ここで，f(i,j)はアドレス(i,j) における濃度値，g(i,j)はアドレス(i,j) におけるグラディエント値である．１画素の計算に１４ステップ必要なため，７ＰＵを並列動作させる．基本的な考え方は混合処理と同じであるので詳細な説明は省略するが，ここでは特に注意すべき点として，絶対値を求める処理について述べる．絶対値を求める処理は，符号をチェックし，負の場合は符号を変え正の場合は何もしないようにすれば求められるが，正と負の場合でステップ数が異なる．このため，画素毎にステップ数が変動することになり，複数ＰＵ間で同期がとれなくなる．そこで，図４．７に示すようにどのパスを通っても同一ステップ数となるように何もしないステッ

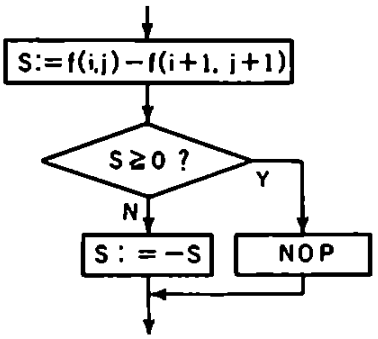


図４．７　条件分岐

プ（ＮＯＰ）を挿入して対応する．

４．４．２　ソフトウェア開発環境

ＲＴＶＰは，１ＰＵ１ステップ当り２７２ビットの幅をもつ水平型のマイクロプログラムで制御するが，ゲートレベルに到るハードウエアとその動作に関する理解が必要であり，プログラミングは簡単ではない．．更に，ＰＵ内の演算要素をパイプライン式にも動作させるために，その前の命令の影響を受ける．またＰＵ群を効率よく動作させることも考慮してプログラムを作成しなければならない．

このための言語として，マイクロアセンブラを作成した．開発したマイクロアセンブラでは，ＰＵ内部を機能ブロックごとに分け，マイクロインストラクションとして定義し直し，命令群を作成した．表４．２に命令群を，図４．８に記述例の一部を示す．ＰＵ内には，多くの演算が含まれているので，並列処理のためには１ステップに多くの命令を記述しなければならない．そこで，ゲート動作の細部や使用しない演算部については，記述しなくても処理系（クロスアセンブラ）が自動的に行うようにして，プログラミングの煩雑さを軽減した．この処理系は，ホストコンピュータ上にインプリメントした．このクロスアセンブラは，ソースからＲＴＶＰ用のオブジェクトを生成する．この際，ＰＵを並列動作させると，複数のＰＵの同じアドレスに同じプログラムがある場合が多いので，同じプログラム部分を抽出し，ブロードキャストモードのパケットに変換して，転送パケット数を減らすようにした．

また，クロスアセンブラと共に，ローダ，コントローラを作成し，プログラムの転送，バンク切り替えなどの制御を行えるようにした．

表4．2　RTVPのマイクロアセンブラ命令群

機　　能	命　令　群
プログラム制御に関する命令	CONT　JMP LDCT　RPCT
映像転送路に関する命令	VTL
画像メモリに関する命令	IM
画像メモリのアドレスを制御する命令	SXY　HXY
映像転送路への出力に関する命令	SD　HD
XP, YP, DP間のデータ交換の命令	MVS　MVH
LUTに関する命令	LUT
乗算器に関する命令	MPI　MPY
ALUに関する命令	LDI　ADI　SBI　SRI ANI　ORI　LDA　ADA SBA　SRA　ANA　ORA

```
* sample program
UNIT2/[IM B & VTL B]
01:LDI D0,4000
02:JMP SFD,02 & LDI X0,0000 & LDI Y0,0000
03:JMP NSHD,03
04:JMP SHD,04
05:ADI X0,0001 & MPY DALU,IM & SXY ALU,ALU & HD LUT & VTL C
06:JMP NSHD,05 & LUT MSP & HXY ALU,ALU & SD LUT & VTL C
07:LDI X0,0000 & ADI Y0,0001
08:JMP NSHD,05
09:JMP NC,08
/
UNIT3/
01:CONT
```

図4．8　プログラム記述例（一部）

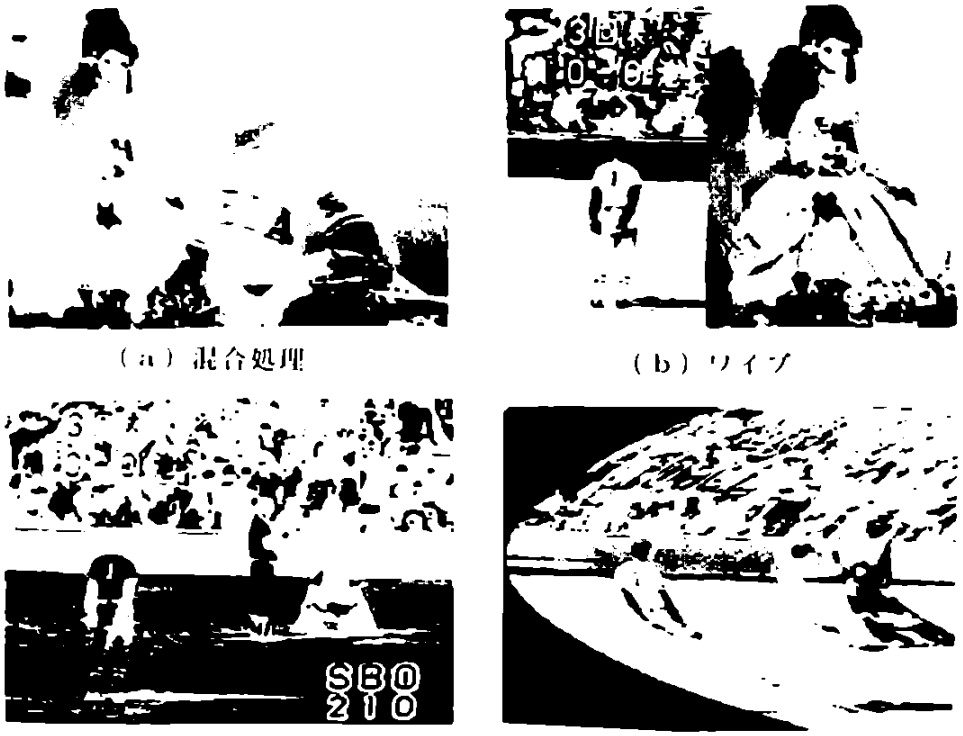
４．５　各種映像信号処理の適用と評価

先にも述べたように，ＲＴＶＰは１個のＰＵにつき１画素当り２ステップの命令を実行でき，ｎ個のＰＵでは１画素当り２×ｎステップに相当する命令を実行することができる．従って，ビデオレート処理するためには，処理の総ステップ数に応じて必要な数のＰＵを割り当てることになる．ＲＴＶＰに，種々の映像信号処理を適用した場合の必要ＰＵ数を，表４．３に示す．なお，混合処理については，４．４．１では並列動作の説明のために冗長な処理形態をとり４ＰＵを使用したが，（４．９）式は，ＰＵ内の乗算器とＡＬＵをパイプライン化して使用することにより４ステップ毎に計算できるため，表４．３では２ＰＵと記述してある．また，動きベクトル検出は，代表点マッチング法[4-7]を適用した場合，アフィン変換は最近傍補間法を適用した場合，テクスチャマッピングは，映像を球体の表面にマッピングした場合に必要なＰＵ数を示す．処理例を図４．９に示す．（ａ）は混合処理，（ｂ）はワイプ処理，（ｃ）はネガポジ反転，（ｄ）は幾何学変換例である．

通常，画像処理装置の性能を評価するのには，①典型的な画像処理のベンチマークプログラムの処理時間で比較する方法[4-8][4-9]，②単位時間や画素

表４．３　各種映像信号処理のＲＴＶＰへの適用

処理内容		処理に必要なＰＵ数
濃度値変換	ゲイン調整	1
	γ補正	1
	しきい値調整	1
複数映像間演算	混合処理	2
	マスキング	2
	横ワイプ	1
近傍画素演算	水平微分	2
	平滑化（２×２）	5
	グラディエント	7
時間軸処理	フレームシンクロナイズ	1
	フレーム間差分	3
	動きベクトル検出	6
幾何学変換	アフィン変換	1
	テクスチャマッピング	5



（ａ）混合処理　　（ｂ）リップ

（ｃ）ネガポジ反転　　（ｄ）幾何学変換

図４．９　処理例

当りの演算回数を評価するMOPS，PIXOSを用いて比較する方法，③演算並列度，画素並列度，近傍並列度，ビット並列度などの評価尺度の積で比較する方法などが提案されている　　か，ビデオレート処理装置であるＲＴＶＰでは，すべての処理が１フレーム当り１／３０秒で完了することが前提であるので，①のように処理時間で評価することができない．また，ＲＴＶＰはＰＥ数に比例して単位時間当りの演算回数（ステップ数）を向上させることができるため，②のような評価法は意味を持たない．③については，アーキテクチャの評価になってもシステムの性能評価とはなりにくい．

一般に，並列処理ではプロセッサ数を増やすことによって処理能力を向上させるか，アーキテクチャに依存して向き不向きの処理があり，プロセッサ数に比例した性能を出すことができないのが普通である．例えば，完全並列方式では近傍データを扱うのには向いているが，幾何学変換を行うためにはプロセッサ間でデータのやり取りが必要になり効率が下がることなどがあげられる．

そこで，ＲＴＶＰの処理能力を評価するために，ＲＴＶＰに適用した各種処理を汎用のコンピュータにも適用して，処理時間を計測し，処理内容の違

いによる処理時間の変化のようすとＲＴＶＰにおける必要ＰＵ数とを比較した．表４．４は，画像のコピー処理を基準にして，ミニコンピュータでの各種映像信号処理の相対処理時間Ｔと，ＲＴＶＰにおける必要なＰＵ数／Ｔ（＝Ｒ）を比較したものである．ノイマン型のコンピュータでは処理速度は処理手順の質よりも処理手順の量に依存しており，処理内容によるばらつきは少なく，汎用性は高いと言える．従って，ＰＵ数の増加率がミニコンピュータにおける処理時間の増加率に近いほど汎用であると言える．評価は，１４．３ＭＨｚ標本化に準じた画素数７５４×４８３の画像に対して行った．入力画像配列ＩＭ１を出力画像ＩＭ２にコピーする処理を基準にして処理時間を示した．仮想記憶によるメモリスワッピングの影響を小さくするために，画像の配列は２個に限定し，２入力必要なものについてはＩＭ１とＩＭ２を入力画像としＩＭ２を出力画像とした．

表４．４　ＲＴＶＰとミニコン（ＭＶ４０００）の処理能力比較

処理内容		ミニコン上の（*）相対処理時間 Ｔ	Ｒ＝ 必要ＰＵ数／Ｔ
濃度値変換	ゲイン調整	１．３	０．８
	γ補正	１．３	０．８
	しきい値調整	１．３	０．８
複数映像間演算	混合処理	２．０	１．０
	マスキング	１．５	１．３
	横ワイプ	１．５	０．７
近傍画素演算	水平微分	１．５	１．３
	平滑化（２×２）	２．４	２．１
	グラディエント	２．６	２．７
時間軸処理	フレームシンクロナイズ	―（**）	―（**）
	フレーム間差分	１．５	２．０
	動きベクトル検出	１．９	３．２
幾何学変換	アフィン変換	１．３	０．８
	テクスチャマッピング	１．８	２．２
画像のコピー		１．０	１．０

（*）画像のコピーを基準（１）とした．
（**）フレームシンクロナイズのシミュレーションは行えない．

表４．４によれば，濃度値変換，複数映像間演算，アフィン変換などは，Ｒ＝０．７～１．３とミニコンピュータと同程度であるのに対し，近傍演算や時間軸処理に関しては，Ｒ＝１．３～３．２とミニコンピュータに比べ処理効率が良くない．

４．６　改善すべき課題

ＲＴＶＰの試作，各種映像信号処理の適用実験から，次の問題点や改善すべき点が明らかになった．

（１）表４．４からもわかるように，近傍演算や時間軸処理では，ミニコンピュータに比べ処理効率が良くない．これは，ＶＴＬあるいはＩＭより２つ以上のデータを同時に取り込んで処理できないためである．この点については，ＡＬＵに使用したマイクロプロセッサＡＭＤ２９０１が２つの入力のうち，一方は内蔵のレジスタに固定されており外部より２つのデータを同時に取り込めないことに起因する．ＲＴＶＰでは，実験システムということで市販のマイクロプロセッサを使用したため実現できなかったが，１度に外部から２つ以上のデータが取り込めるよう演算部とすれば，

スムージング　　５ＰＵ　→　３ＰＵ
グラディエント　７ＰＵ　→　４ＰＵ
フレーム間差分　３ＰＵ　→　２ＰＵ
動きベクトル　　６ＰＵ　→　４ＰＵ

など，かなり改善することができる．演算部と画像の入出力部及び画像メモリは密につながっている必要がある．

（２）ＶＴＬが３本では少ない．例えば，図４．１０に示すように処理形態を考える．すなわち，

①映像入力Ａを複数のＰＵで処理して，処理結果ＣをＶＴＬに出力する．
②映像入力Ｂを複数のＰＵで処理して，処理結果ＤをＶＴＬに出力する．
③ＣとＤを複数のＰＵで処理して，処理結果ＥをＶＴＬに出力する．
④ＡとＥを複数のＰＵで処理して，処理結果ＦをＶＴＬに出力する．
⑤ＢとＥを複数のＰＵで処理して，処理結果ＧをＶＴＬに出力する．
⑥ＦとＧを複数のＰＵで処理して，処理結果ＨをＶＴＬに出力する．
⑦Ｈを最終の処理結果として，映像出力に出力する．

この場合，Ａ～Ｈの８種の映像をＶＴＬに載せる必要があるが，うまく割り当てても，最低５本のＶＴＬが必要である．この主な原因は，映像を後段のＰＵで使うために，途中のＰＵが使用しないにもかかわらず，ＶＴＬ上を転送しなければならないからである．処理形態が複雑になると，さらに多くのＶＴＬが必要である．試作では白黒映像であったが，カラー化するとＲＧＢ

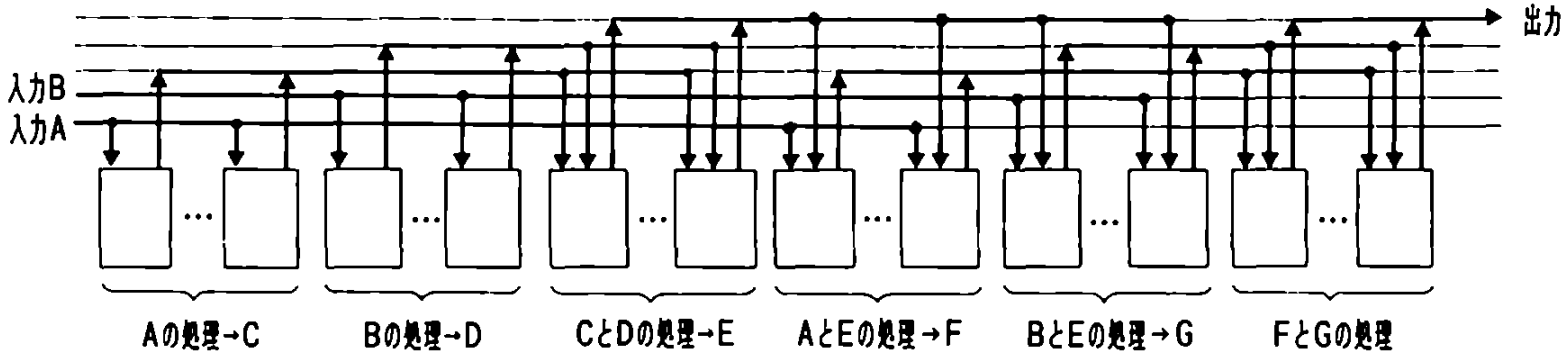


図4．１０　多くのＶＴＬが必要な場合

コンポーネントの場合３倍のＶＴＬが必要となり，回路規模が大きくなってしまう．ＶＴＬの機能を単純な転送路ではなく，可変構造を有するものにしないと使用効率が上がらない．

（３）ＰＵ内ではフィードバックループを構成できるが，ＰＵ間ではフィードバックループを構成できない．３本のＶＴＬのうち，出力の１本を入力の無い１本に接続すれば，フィードバックループが一応構成できる．しかし，２つのＰＵ間でフィードバックループを構成したい場合にも余分なパスを通す必要がある．（２）と同様，柔軟性のあるＶＴＬが望ましい．

（４）絶対値演算は，グラディエントや動ベクトル検出などに頻繁に使用されるために，ソフトウェアではなくハードウェアで実現した方が高速化できる．同様なものに，ＮＡＭ（Non Additive Mixture）などにおける最大値，最小値演算がある．

（５）演算のオーバーフローをチェックして，白黒反転を生じないようにクリッピング動作をさせようとすると，余分のプログラムステップが必要となり，リアルタイム性に欠ける．各演算器にクリッピング回路を持たせるべきである．

（６）乗算器は１６ｂｉｔ×１６ｂｉｔで３２ｂｉｔ出力のものであるが，乗算結果を上位ワードと下位ワードの２回に分けて読み出す方式のＬＳＩしか入手できなかった．その他の演算器は１６ｂｉｔ処理のため，どちらか一方を使うことになる．しかし，ゲイン調整を行う場合には，１．０前後の係数を画像データに乗ずる必要があるが，下位ワードを使うと１．０，２．０など整数倍しか扱えず，上位ワードを使うと１未満の小数が扱えるだけで，１．２倍などの係数が精度よく扱えない．このために，１．０前後で小数部を含む係数が扱えるような出力モードがあると便利である．

（７）レジスタがＡＬＵのみに付属しているために，乗算器の係数を設定するために，ＡＬＵを動作させなければならず，無駄が生じる．能力を十分生かすためには，各演算器にレジスタを設けるべきである．

（８）アドレス計算が１６ｂｉｔ精度では不足である．これは，２．２の検討でも明らかである．ＲＴＶＰでは，アドレス演算器ＸＰ，ＹＰにデータ演算器ＤＰと同じ仕様の演算回路を使用したためである．アドレス計算に適した演算回路と精度とすべきである．

（９）映像信号を扱うために，ＨＤ，ＶＤなどの同期パルスをチェックする必要があるが，ＡＬＵのフラッグをチェックしようとすると同期パルスがチェックできない．交互にチェックすることで対応すると，１クロック分の遅れが生じる．少なくとも２つのフラッグが同時にチェックできるようにすべきである．

（１０）ＶＴＬの制御プログラムがＰＵのプログラムと混在して記述されているために，プログラム構造が分かりにくい．処理そのものの記述とＰＵ間の結合に関する記述は，別々に行える方がよい．
（１１）ホストコンピュータとＰＵ間は，ホストコンピュータのシステムバスを延長して接続したが，バスはあまり延長することができない点と接続本数が多くなる点から，ＰＵ数の増大には限度がある．ＬＡＮ（Local Area Network）などの採用が必要である．
（１２）ＰＵ数が増えると，全てのプログラムを１フィールド内で送れなくなる．前もって，ロードしておく必要がある．この際，１ｋｗのプログラムメモリの容量では不足である．同様のことがルックアップテーブルでも言える．

ＲＴＶＰの試作により，上記の問題点，改善すべき点などの知見が得られ，ビデオレート映像信号処理システムの構成上のキーとなるノウハウが蓄積できたため，これらをもとに実用システムを作ることにした．

［５］実用システム
Ｐｉｃｏｔ－ｓｙｓｔｅｍ

５．１　設計指針

ＲＴＶＰの試作結果を踏まえて実用システムＰｉｃｏｔ－ｓｙｓｔｅｍを開発した．Ｐｉｃｏｔ－ｓｙｓｔｅｍを開発するに当り，以下の目標を設定した．

ＮＴＳＣ信号を対象とする．信号形式は，コンポジット信号（ＮＴＳＣ信号）あるいは，コンポーネント信号（[R.G.B]，[Y.R-Y.B-Y]，[Y.I.Q]）である．コンポジット，コンポーネントの両方を扱うために，標本化周波数は，１４．３ＭＨｚを採用する．プロセッサの処理速度もこの標本化周波数と同じにする．すなわち，サイクルタイム７０ｎｓである．演算語長は１６ビットとする．この仕様であれば，放送に使用しても十分な画質である．

この基本システムに加え，実用的なシステムとして放送用ビデオスイッチャを開発し，放送現場で稼働させ検証を行う．このために，ビデオ入出力は，最低必要な４～８程度の入力数，２～３程度の出力数に対応できるようにする．また，プロセッサ数は，通常使われている特殊効果などを含む３効果列程度に対応できるように，２００台程度用意する．このために，主要部は，ＬＳＩ化して小型化を進める．さらに，専用の操作卓とアプリケーション・ソフトウェアを作成する．

５．２　アーキテクチャ

５．２．１　概要

Ｐｉｃｏｔシステムは，図５．１に示すように多段に接続されたクラスタ群，ホストコンピュータ（ラップトップ・パーソナルコンピュータＪ３１００），コントロールパネル（制御卓）から構成される[1]．クラスタ間は，単方向性バス１６系統で接続されている．初段のクラスタの入力に映像信号を入力し，クラスタ群で処理し，最終段のクラスタの出力に処理結果を出力する．映像入力，映像出力とも最大１６系統接続できる．標本化周波数は，１４．３ＭＨｚである．クラスタは，図５．２に示すように１６台の同一仕様のプロセッサを中心に構成される．１６台のプロセッサの入出力４８系統（各プロセッサは，２入力１出力になっている）及び，クラスタの外部入力１６系統，外部出力１６系統が，ネットワークで結合されている．各系統は，同期情報（水平同期ＨＤ，垂直同期ＶＤ，フレーム同期ＦＰ）を多重化できるように，１６ｂｉｔのデータに同期ビットを付加した１７ｂｉｔ幅である．クラスタ内には，コントローラがあり，プロセッサとネットワークを含むク

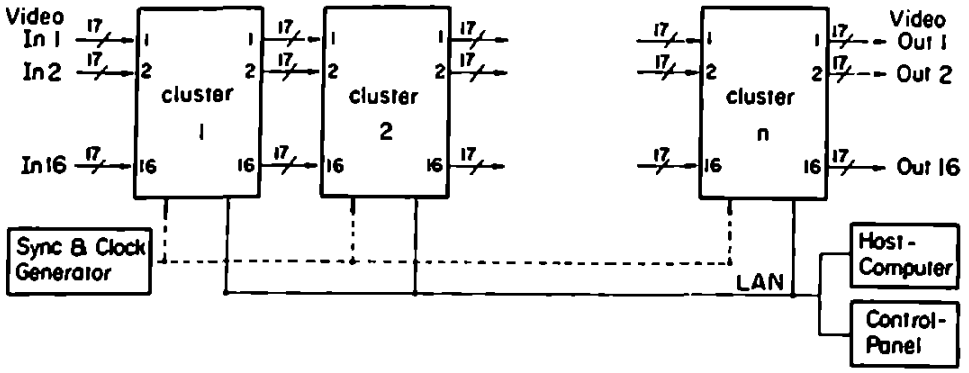
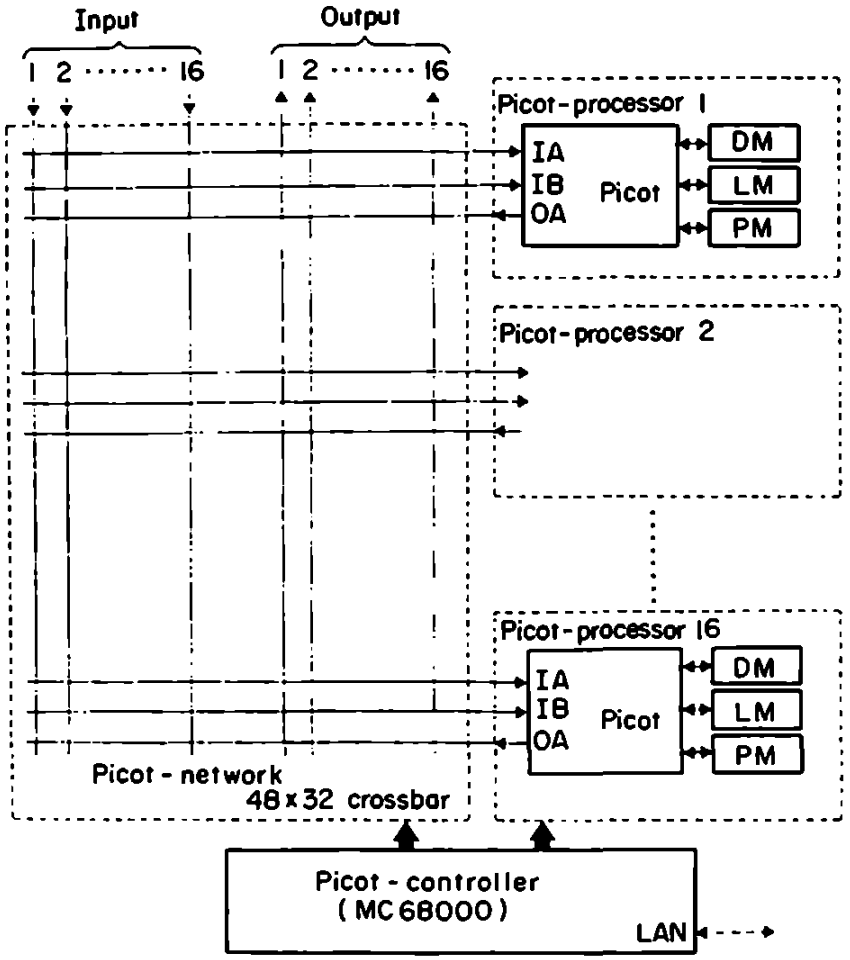


図5. 1 Picot-systemの構成図



DM: Data Memory (256K x 16b) PM: Program Memory (4K x 32b)
LM: Line Memory (1H x 16b)

図5. 2 Picotクラスタの構成図

ラスタの状態を管理する．

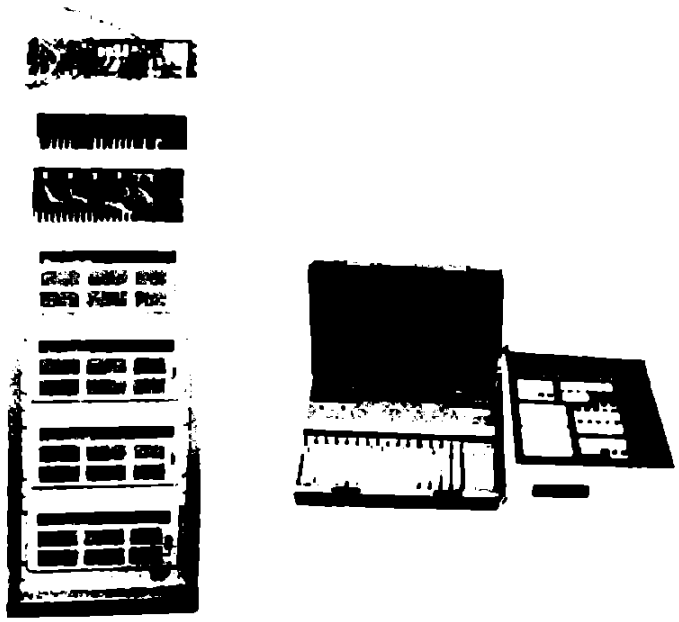
Ａ／Ｄ，Ｄ／Ａは，通常８ビット程度であるので，データのダイナミックレンジを確保でき，かつ演算精度も保つことができるように，１６ビットの上位から５ビット目から割り当てるようにした．これにより，データレンジとして±８倍までの余裕があり，かつ小数部に４ビットの精度を保つことができる．

コントロールパネルは，ジョイスティック（Ｘ，Ｙ，Ｚの３軸型），スイッチ群などから構成されている．ホストコンピュータでは，主にプログラムの開発とシステム全体の制御を行う．クラスタ群，ホストコンピュータ，コントロールパネルは，すべて１本のＬＡＮで接続されており，ホストコンピュータあるいはコントロールパネルからクラスタ群に，プログラム，パラメータ，コマンドを送り，これをコントローラが解析し，プロセッサやネットワークを制御することによって，各種の動画像処理を行う．

表５．１に主な仕様を，図５．３に４クラスタ構成のシステムの写真を示す．

表５．１　Ｐｉｃｏｔ－ｓｙｓｔｅｍの主な仕様

映像入力	標準ＴＶ（ＲＳ１７０Ａ）１６系統 （[NTSC]，[R，G，B]，[Y，R-Y，B-Y]）など
映像出力	標準ＴＶ（ＲＳ１７０Ａ）１６系統 （[NTSC]，[R，G，B]，[Y，R-Y，B-Y]）など
標本化周波数	１４．３ＭＨｚ
データ語長	１６ｂｉｔ
処理速度	７０ｎｓ／ｐｉｘｅｌ，３３ｍｓ／ｆｒａｍｅ
プロセッサ数	１６×ｎ　（ｎ：クラスタ数）
制御方式	マイクロプログラム制御
処理機能	点演算（濃度値変換，しきい値処理など） 色処理（色補正，クロマキーなど） 複数映像間演算（ミックス，ワイプなど） 近傍画素演算（輪郭強調，ノイズ除去など） 幾何学変換（アフィン変換，透視変換，マッピングなど） 時間軸処理（フレーム間処理，動き検出など）



左から，本体(A/D，D/A，クラスタ)，ホストコンピュータ，コントロールパネル

図５．３　Ｐｉｃｏｔ－ｓｙｓｔｅｍの外観写真

５．２．２　並列処理方式

Ｐｉｃｏｔ－ｓｙｓｔｅｍでは，３．２で提案したConfigurationＢ を採用した．ＲＴＶＰで採用したConfigurationＡ では，①フィードバックループが構成しずらい，②前段のプロセッサから後段のプロセッサにデータを渡すためだけに多くのＶＴＬを使用するなどの欠点を回避するためである．

ConfigurationＢ では，１６個のプロセッサをひとまとまり（クラスタ）にして，この中では任意の接続ができるようにし，このクラスタを多段に接続して構成する．クラスタ間は，１６本のバスで接続されている．これにより，クラスタをまたがって，複数の映像入力を並行して供給したり，処理結果の映像や処理途中の映像も出力でき，処理能力をあまり犠牲にすることなく，パイプライン処理や並列処理を実現することができる．

５．２．３　メモリ構成法

画像メモリは，３．３で述べたように，各プロセッサにメモリをもたせる方式とした．必要なデータの配置は，処理の内容によって前もってわかっているので，処理に先立ちそれぞれのプロセッサのメモリに格納しておく．プ

ロセッサが必要とするデータが複数のプロセッサで同じであることが起こりうるが，その場合は，別々のプロセッサに同じ内容のデータが確保されることになる．この方式は，メモリアクセス競合や通信手続きが不要のため，プロセッサ数に比例した処理能力向上が望める．画像メモリの容量は，幾何学変換を行うのに最低必要な１フィールド（インタレース走査しているＮＴＳＣ信号では，１フレームの半分の容量，２５６ｋｗ）とした．

データメモリへの書き込みと読み出しはＲＴＶＰと異なり同じアドレス演算器を使用する．従って，書き込みと読み出しは同時に行えない．これは，標本化周期が７０ｎｓのために同時に書き込みと読み出しを行うためにはサイクルタイムが最悪でも３５ｎｓのメモリが要求され，コストアップにつながるからである．これは，幾何学変換する場合に，同じメモリに書き込みながら読み出すと，読み出しより早くなる追越しが起きて，１つの画面に異なる時刻の画面が合成されて表示される現象を防ぐ効果もある．

５．２．４　同期方式

３．４で述べたように，多数のプロセッサを使用するマルチプロセッサシステムにおいては，処理過程でさまざまな遅れが生ずるので，プロセッサ間の同期対策が必要である．

Ｐｉｃｏｔ－ｓｙｓｔｅｍでは，次に示す３つの方法を採用した．

（１）各プロセッサに，最大７画素の可変遅延要素と，最大１ライン（９１０画素）の可変遅延要素（ラインメモリ）及びフィールドメモリをもたせた．これをソフトウェアで制御することにより，ずれを吸収できるようにした．また，フィールドメモリを位相合わせのために使用するのは最終処理出力のみとした．

（２）システム同期信号を遅らせるために，クラスタ内の同期信号分配回路に可変遅延回路（最大１０２４クロック）を設けた．システム同期信号は，本来プロセッサ毎に変えるのが望ましいが，処理のやり方によって供給すべき同期信号を変える必要があり現実的でない．そこで，クラスタ内は同じシステム同期信号とした．このため，プロセッサによっては，映像とシステム同期が完全には一致しない場合が生じ，有効画面が欠落することがある．これを避けるために，水平同期信号の幅を同期信号規格よりも狭くした．Ｐｉｃｏｔ－ｓｙｓｔｅｍにおける処理の遅延量Ｄは，$(4+2\times e)\times p+2)\times c$である．ここで，ｃはシステムを構成するクラスタ数（図５．３の例では，４台），ｐは処理に割り当てるプロセッサ数，ｅは処理に割り当てるプロセッサ内の演算器数である．従って，１クラスタ当りの最大の遅延量Ｄは１６２（ｅ＝３，ｐ＝１６，ｃ＝１の時）である．このため，１段目のクラ

スタに対して，２段目は１６２クロック，ｎ段目は１６２＊（ｎ－１）クロック遅れたシステム同期信号を供給すればよい．通常，全ての演算器を使用することや，全てのプロセッサをパイプライン接続して使用することは希であるので，経験的には，Ｄは６６（ｅ＝２，ｐ＝８，ｃ＝１の時）でも，問題はないと考えている．５．６．１で述べるビデオスイッチャではこの値に設定している．

（３）映像を同期情報付きで送るために，プロセッサ間及びクラスタ間の接続を，１６ビットの映像データに同期ビットを付加した１７ビット幅にした．同期情報は，プロセッサのプログラム制御回路に入力でき，これにより，映像と同期に合致してプロセッサを制御することができる．

５．２．５　プログラムロード方式

３．５で述べたように，プログラムのロードは垂直帰線期間内に行う．しかし，ＲＴＶＰのように毎垂直帰線期間に強制切り替えを行うことはせず，プロセッサがＲＥＳＴＡＲＴと名付けられた命令を実行した場合のみに行うようにした．これは，フィールドをまたがる処理，例えばフレーム単位の処理やフィールドとは関係なく処理が完了するまで行うタイプの処理に対応するためである．

実用システムでは，数百台のプロセッサを接続することになるため，１プロセッサ当り２００ステップのプログラムとしても，データ量は１００ｋｂｙｔｅを越える．これをロードするためには，１フィールド（１／６０秒）全てを費やしたとしても，数Ｍｂｙｔｅ／ｓｅｃ以上の転送レートが要求される．この転送は，パラメータの計算など他の処理を行わねばならないホストコンピュータには処理しきれない．そこで，複数種類のプログラムを前もってプロセッサにロードしておき，この中からどれを実行するのかを指示する方式とした．この指示のためには，実行したいルーチンの先頭アドレスとともにパラメータを渡す必要がある．このデータのやり取りのために，プロセッサ内に専用のインタフェース・レジスタを２セット（各８ｗ×１６ｂｉｔ）設けた．これにより，メインプログラムが，サブルーチンを呼ぶような形式で，多数のプロセッサを制御することができる．

５．３　ハードウェア構成

５．３．１　プロセッサ

プロセッサは，１４．３ＭＨｚ（サイクルタイム７０ｎｓ）の速度で映像信号を１６ｂｉｔ精度で処理する．構成を図５．４に示す．入力ポートを２系統，出力ポートを１系統もつ．データ幅は，同期情報を送受するためのビ

ットも含め１７ｂｉｔである．ＡＬＵ（Arithmetic Logic Unit），乗算器，アドレス演算回路ＨＡＵ（Horizontal Addressing Unit）及びＶＡＵ（Vertical Addressing Unit），シーケンサ等からなる．全ての演算は，１サイクル（７０ｎｓ）で実行できる．これらは，複数バスにより接続されており，パイプライン動作や並列動作が可能である．

（１）演算回路

ＡＬＵは，１６ｂｉｔ（２の補数）精度で処理する．加減論理演算機能に加えて映像信号処理に不可欠な絶対値，最大値，最小値演算機能を有する．また，オーバーフローに対しては，自動的にクリップがかかる．これにより，視覚的に目だつ白黒反転を防ぐことができる．

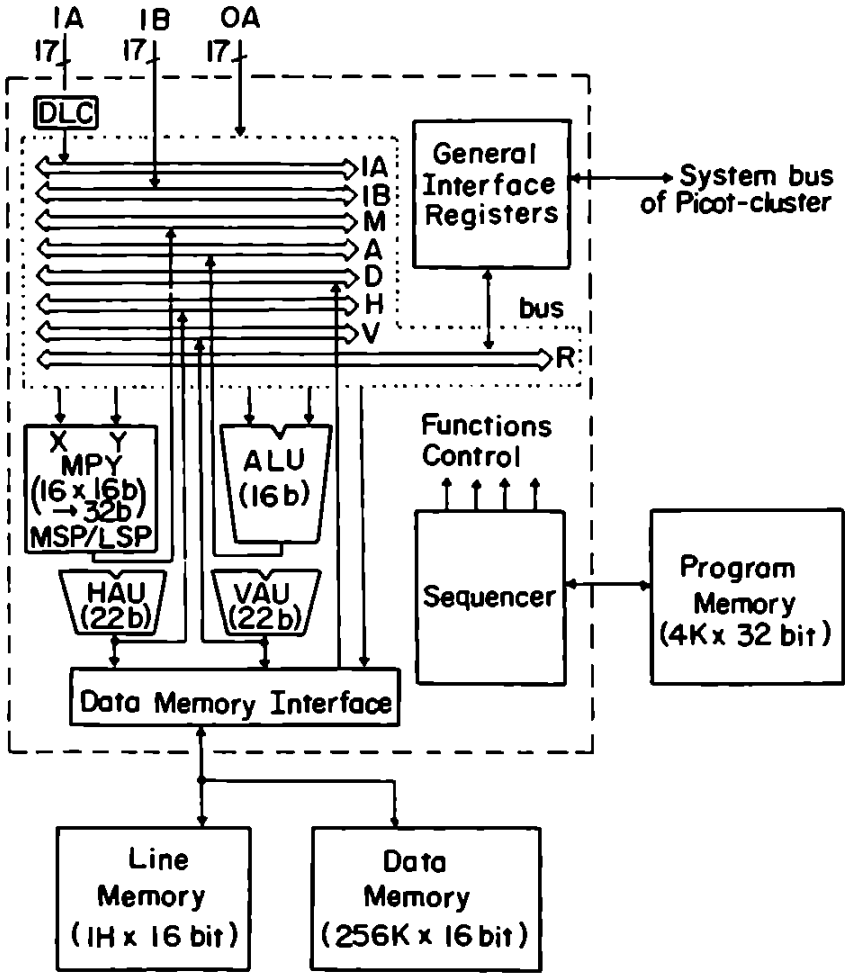
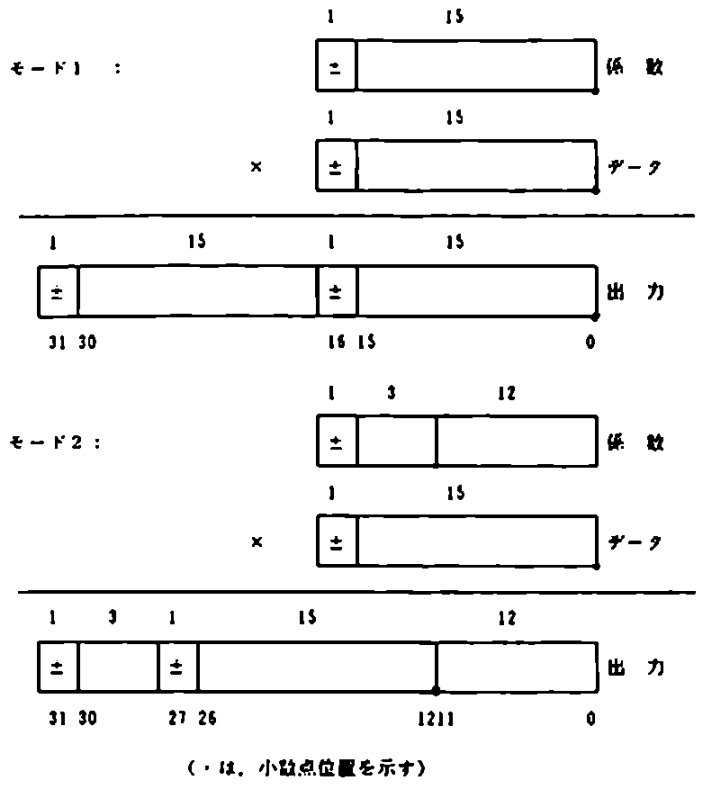


図５．４　プロセッサの構成図



（・は，小数点位置を示す）

図５．５　乗算器の演算モード

乗算器は，１６ｂｉｔ×１６ｂｉｔで行い１６ｂｉｔの出力を得る．２次のＢｏｏｔｈのアルゴリズムを採用しており，ＡＬＵと同様１サイクルで求めることができる．演算モードは，図５．５に示すように通常の乗算モード１に加え，係数の絶対値を最大８に抑えて小数部に１２ｂｉｔ割り当てることにより，演算時の有効桁数が多くなるモード２を用意した[11]．

（２）アドレス回路とデータメモリ

ＨＡＵとＶＡＵは，アフィン変換が可能なアドレス演算回路である．図５．６に示すように，同一の回路構成をしており，それぞれ，加減算回路とコンパレータ２個から成る[11]．２．２で検討を加えた図２．１に初期値レジスタhint2.vint2 を加えた構成をしており，それぞれ（２．１）（２．２）式の演算を毎クロック行うことができる．演算，レジスタの選択は，マイクロプログラム制御のシーケンサで行う．しかし，同じ演算を繰り返す場合は，命令を繰り返すように記述しなくても演算は自動的に継続するようになっている．更に，演算と同時に比較を行うためにコンパレータを２つ設けた．このコンパレータは，前もって比較値をセットしておけば毎クロック比較しフラッグを出力する．コンパレータ１の場合，ＨＡＵ，ＶＡＵそれぞれのデー

タが比較値レジスタhcmp1.vcmp1 未満のときにフラグhauf1.vauf1 が立ち，コンパレータ２の場合，比較値レジスタhcmp2.vcmp2 を越えるときそれぞれフラグhauf2.vauf2 が立つ．これらのフラグは，シーケンサの判定条件に設定できる．この条件と同期信号（水平同期ＨＤ，垂直同期ＶＤ，フレームパルスＦＰ）を同時にチェックでき，優先度も任意に設定できるため，２．２．２．３で述べた基本波形発生や幾何学変換などにおいて，遅滞なく処理を進めることができる．

ＨＡＵ．ＶＡＵの演算語長については，２．２．２．３の検討結果があり，一方ハードウェア規模を小さくし，かつ演算速度を１サイクル７０ｎｓ以下に抑える必要があったため，２２ｂｉｔとした．これは，標準ＴＶ信号を１４．３ＭＨｚで標本化した場合に１画素単位の精度を満たす語長である．標準的には，整数部を１６ｂｉｔ，小数部を６ｂｉｔとしている．

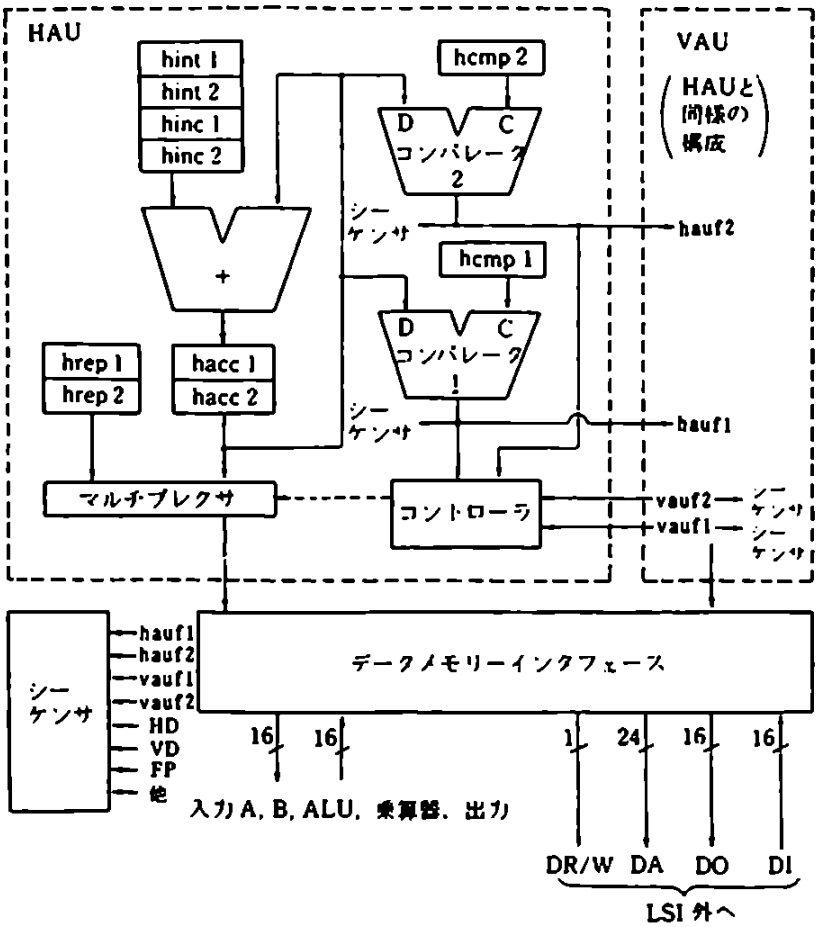


図５．６　ＨＡＵ，ＶＡＵの構成図

ＨＡＵとＶＡＵを使用すると，演算と並行してデータメモリへのアクセスが可能である．データメモリは，２５６Ｋ×１６ｂｉｔの容量（１０２４×２５６画素の画像に相当）で，１サイクルで書き込み又は読み出しができる．

ＨＡＵとＶＡＵの各２２ビットの演算出力から，アドレス幅１８ｂｉｔ，容量２５６ｋｗのデータメモリにアクセスするためには，アドレスの変換が必要である．ＮＴＳＣ信号用フィールドメモリを想定すると，横１０２４画素×縦２５６画素となるので，ＨＡＵから１０ｂｉｔ，ＶＡＵから８ｂｉｔ抜き出して連結し１８ｂｉｔとすればよい．この方式では，上位ビットが切り捨てられるために縮小時に，ＨＡＵ，ＶＡＵ内の演算のオーバーフローよりも先に，出力アドレスが循環し同じ画像が繰り返し出力されてしまう．これを避けるためには，ＨＡＵ，ＶＡＵからデータメモリへの出力をそれぞれ０～１０２３，０～２５５の範囲に抑えなければならない．これをプログラムで行っては複数のプログラムステップを必要とするので，ハードウェア的にクリップされるようにした．すなわち比較値レジスタhcmp1.hcmp2.vcmp1.vcmp2 の値を有効画面の端に設定し，すべてのフラグhauf1.hauf2.vauf1.vauf2 が０となる有効画面内ではhacc，vaccをそのまま出力し，それ以外の条件となる有効画面外にはみ出した時，hrep1.vrep1 の値が出力されるモードを設けた．hrep1.vrep1 の画素に０を設定しておけば，繰り返しはなくなり黒が出力されることになる．この他に，

① 全てのフラグhauf1.hauf2.vauf1.vauf2 が０の時，hrep1.vrep1，それ以外の時は，hrep2.vrep2を出力する．（マスキングパタンを発生させるために使用する．），

② hauf1=1の時hrep1，hauf2=1の時hrep2，それ以外の時haccを出力する．

ＶＡＵも同様に動作する．（ＨＡＵ，ＶＡＵ独立のクリップで，データメモリをルックアップテーブルに使用する際にテーブルの参照可能範囲を越えないように制限するために使用する．）

などのモードも用意した．

なお，データメモリは，ルックアップテーブルとしても使用でき，１６ｂｉｔ入力１６ｂｉｔ出力の非線形演算器が４個用意できる．テーブル内容をロードするためには，後述のコントローラを使用する．データメモリは，コントローラのメモリ空間にマッピングされており，書き込み読み出しは容易である．また，ルックアップテーブルは，幾何学変換の際の４点補間にも使用できる．このために，ＨＡＵ，ＶＡＵの小数部を連結して，データメモリにアクセスするモードも用意した．（２．２３）式を変形すると，

$$d=(1-q)(1-p)\cdot d(i,j)+(1-q)p\cdot d(i+1,j)+q(1-p)\cdot d(i,j+1)+qp\cdot d(i+i,j+1) \qquad (5.1)$$

のようになる．補間係数(1-q)(1-p)，(1-q)p，q(1-p)，qpを前もって計算しデータメモリにロードしておき，これをルックアップテーブルとして参照するようにすれば，プロセッサ1台で（5．1）式の各項を計算できる．

この他，複数のプロセッサを使ってアフィン変換より複雑なアドレス発生を行いたい場合がある．このために，他のプロセッサからアドレスを受け入れられるようにＨＡＵの出力の代わりに入力Ａを，ＶＡＵの代わりに入力Ｂをデータメモリのアドレスに接続するモードも用意した．

（3）入出力回路

2系統ある入力のうち1系統は，水平方向の画素間演算が容易になるように最大7画素の遅延がプログラムで可変にできるようになっている．また，垂直方向の画素間演算のためには，最大1ラインまで遅延量を可変にできるラインメモリを有している．出力は，ＡＬＵ，乗算器，ＨＡＵ，ＶＡＵの演算結果，データメモリ，ラインメモリの出力に加え，ＡＬＵの演算フラッグを直接出力することができる．これにより，例えばあるプロセッサでＡ画像とＢ画像を比較したフラッグを画素単位で出力すると，次のプロセッサでは，このフラッグによって処理を画素単位に切り換えることができ，プロセッサ間で連動したプログラム制御を行うことが可能である．なお，入出力は，5．2．4で述べたように，同期情報を付加するために１７ｂｉｔ幅になっている．

（4）シーケンサ

プロセッサは，内蔵のシーケンサによってマイクロプログラム制御される．シーケンサは，マイクロプログラムメモリの内容をデコードして，各処理ユニット，入出力，及びシーケンサ自身を制御する．プログラムは，４ｋ×３２ｂｉｔの容量をもつプログラムメモリに格納する．本プロセッサは，画像，特に映像信号を主対象としているため，映像信号処理をかなり意識した制御方式を採用しているが，これについては，5．3．2で述べる．

（5）コントローラとのインタフェース

プロセッサ内には，各演算器内にあるレジスタとは別にＬＳＩ内で共通に使用できるインタフェース・レジスタが2セット（各８ｗ×１６ｂｉｔ）ある．このレジスタは，コントローラとのインタフェースも兼ねており，レジスタをバッファとして，データなどの送受を行うことができる．これについては，5．3．3で述べる．

なお，プロセッサは，プログラムメモリ，データメモリ，ラインメモリを除いたほとんどの部分をＣＭＯＳゲートアレイを用いて，ＬＳＩ化した．このＬＳＩについては，5．3．6で述べる．図5．7にプロセッサ・ボードの写真を示す．約２０ｃｍ×１０ｃｍの大きさである．

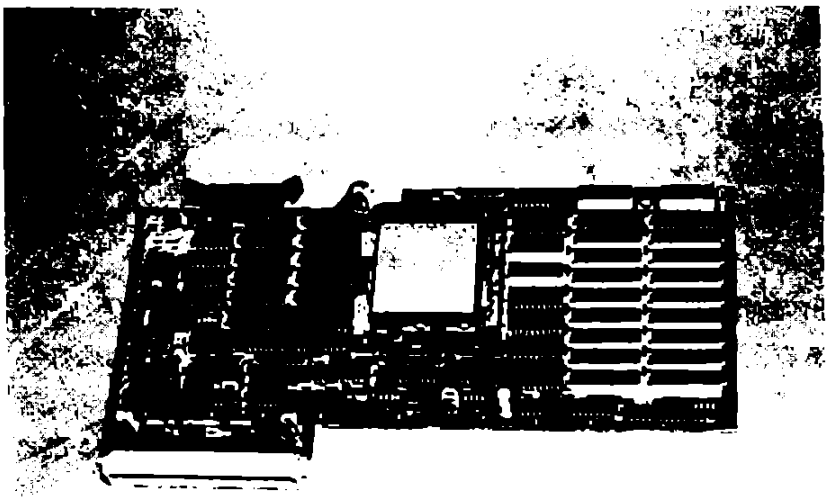


図5．7　プロセッサの外観写真

5．3．2　プロセッサの制御機構

本プロセッサは，画像，特に映像信号を主対象としているため，映像信号処理をかなり意識した制御方式を採用している[7]．

プロセッサは，前述のように多くの処理ユニットを有しており，これら全てを一度に動作させようとすると，１００ｂｉｔ幅以上のマイクロ命令が必要となる．プログラムメモリを外付けとするためには，マイクロ命令用に多くのピンを割り当てることはできない．しかし，ピン数を抑えるためにマイクロ命令のビット幅を抑えると，全ての演算器を同時には有効に使えなくなり，効率が下がる．ここに，相反する要求があった．

そこで，次に示す手段を講じた．

（１）映像信号処理においては，『フレーム内（あるいはライン内）の全ての画素に対して，同じ処理を行う場合が多い』ことに着目し，一旦セットした演算器の命令は，新たに演算器に関する命令がセットされるまで，その演算器は，同じ命令の処理を続けるようにした．

（２）映像信号処理においては，『演算結果フラグや同期信号などの複数の条件をチェックしながら演算を行う場合が多い』ことに着目し，一度に最大２つの条件をチェックでき，これに基づいて分岐するようなシーケンス命令とした．

（３）また，同じ処理を続けることが多い点を踏まえ，通常のマイクロプロセッサが有するＣＯＮＴＩＮＵＥ，ＪＵＭＰ命令に加え，自番地にループす

表5．2　インストラクション・セット

標準の命令形式（32bit）

シーケンス命令(16bit)	ファンクション命令(16bit)

シーケンス命令（16bit）

ＲＪＣ　CC1のときjump，　CC2のときcontinue，それ以外repeat
ＲＣＪ　CC1のときcontinue，CC2のときjump，　それ以外repeat
ＲＪＪ　CC1のときjump1，　CC2のときjump2，　それ以外repeat
ＣＪＪ　CC1のときjump1，　CC2のときjump2，　それ以外continue

・jump1，jump2は，それぞれ別のジャンプアドレスが記述できる．
・CC1，CC2を設定しないことも可能である．従って，

Ｒ　　無条件repeat
Ｃ　　無条件continue
Ｊ　　無条件jump
ＲＪ　CC1またはCC2のときjump，　それ以外repeat
ＲＣ　CC1またはCC2のときcontinue，それ以外repeat
ＣＪ　CC1またはCC2のときjump，　それ以外continue

も記述可能である．

ファンクション命令（16bit）

ＮＯＰ	何もしない
ＡＬＵ	ＡＬＵに関する命令
ＭＰＹ	乗算器に関する命令
ＨＶＡＵ	ＨＡＵ，ＶＡＵに関する命令
ＤＭＩ，ＤＭＢ	データメモリに関する命令
ＤＬＣ，ＴＭＣ	ライン遅延と画素遅延に関する命令
ＯＵＴ	出力に関する命令
ＭＧＳ，ＭＧＧ，ＭＦＧ	演算レジスタ及び汎用レジスタ間のデータ転送命令

例外の命令形式（32bit）

ファンクション命令(32bit)

シーケンス命令（0bit）
記述されていないが，continueを実行する．

ファンクション命令（32bit）

ＬＤＲ　レジスタにイミーディアット値をロードする命令
ＬＤＡ　ＩＡＭにアドレステーブルをロードする命令
ＬＤＰ　ＩＰＭにアドレステーブルをロードする命令

る命令ＲＥＰＥＡＴを用意した．

（４）シーケンサを１４．３ＭＨｚで動作させるため，３段のパイプライン構造としたが，条件分岐が発生した場合にも，パイプライン動作が崩れて処理が中断しないようにする機構を加え，実時間処理を可能にした．

以上の手段により，実時間性を損なうことなく，マイクロプログラム幅を，３２ｂｉｔとすることができた．表５．２に，命令表を示す．全ての命令が，３２ｂｉｔ幅の１ワード命令である．ほとんどの命令が１６ｂｉｔ幅のシーケンス命令と１６ｂｉｔ幅のファンクション命令からなる．以下では，上記の（１）～（４）を実現しているシーケンサの動作について説明する．

シーケンサは，図５．８に示す構成をしている．通常，ＰＣ（Program Counter）から与えられたプログラムアドレスをＡＬ（Address Latch）でラッチする．ラッチされたアドレスは次のサイクルで，プログラムメモリにフェッチされ，ＩＬ（Instruction Latch）にラッチされる．ラッチされたプ

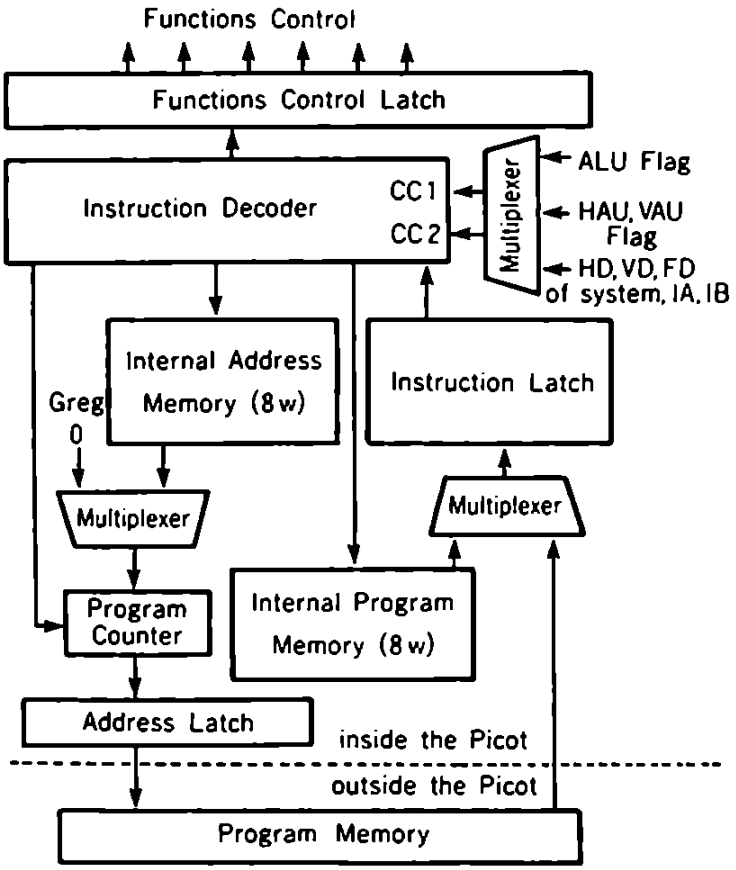
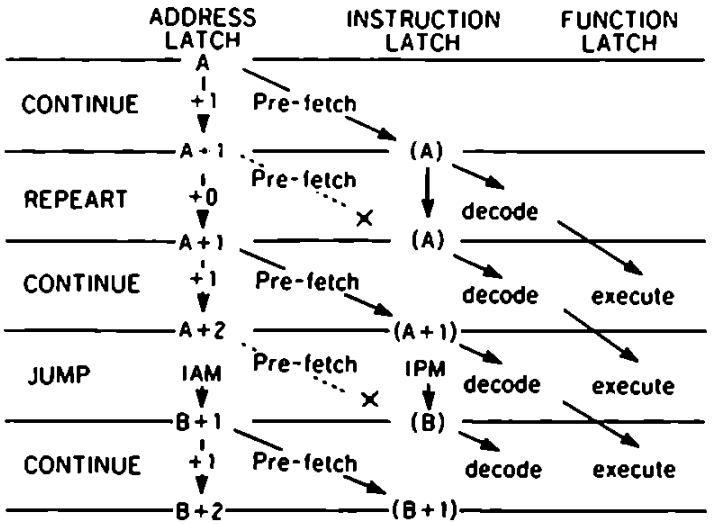


図５．８　シーケンサの構成図

ログラムは，次のサイクルでＩＤ(Instruction Decoder)によりデコードされ，ＦＣＬ(Function Control Latch)でラッチする．この次のサイクルが実行サイクルである．すなわち，（４）で述べたように，フェッチ，デコード，実行の３段のパイプライン構造である．ＦＣＬ(Function Control Latch)は，以前に実行した全ての演算器（ＡＬＵ，乗算器，ＨＡＵ，ＶＡＵなど）の入出力や演算機能などの制御情報を保持している．例えば，ＡＬＵに関するマイクロ命令がデコードされた場合は，ＡＬＵに関する部分のＦＣＬが書き換えられるのみで，そのほかの演算器に関する部分のＦＣＬは前の状態が保持されている．このため，（１）で述べたように保持されている制御情報で常に全ての演算器を動作させることができる．

一般にパイプライン構造を採用すると，条件分岐が発生した場合，パイプラインが崩れ処理が中断する．その結果，映像信号に妨害を与えてしまう．そこで，条件分岐が発生しても処理が中断しないように，ＩＰＭ(Internal Program Memory)と ＩＡＭ(Internal Address Memory) をシーケンサ内に設けた．これらは，ＬＳＩ内のメモリがＬＳＩ外のメモリに比べ高速にアクセスできることを利用したもので，ＩＰＭにはジャンプ先の命令を，ＩＡＭにはジャンプ先＋１のアドレスを格納しておく．それぞれ８ワードの容量があり，ジャンプ処理に先立つ垂直，水平帰線期間にあらかじめ格納しておく．

これらの動作について，図５．９に示す．命令がＣＯＮＴＩＮＵＥの場合



IAM : Internal Address Memory
IPM : Internal Program Memory

図５．９　シーケンサの動作フロー

は，ＰＣは＋１され，pre-fetch がそのまま生き，ＩＬに命令がラッチされる．ＲＥＰＥＡＴ命令の場合は，ＰＣはホールドされ，ＩＬにはpre-fetchされた命令を放棄して前サイクルの命令がホールドされる．ＪＵＭＰ命令の場合は，ＩＡＭからジャンプアドレスをＰＣにロードし，ＩＬにはpre-fetchされた命令を放棄してＩＰＭからマイクロ命令をロードする．これにより，ジャンプしても，パイプライン制御がくずれない．

また，ＩＤの条件入力にはＡＬＵ，ＨＡＵ，ＶＡＵのフラッグ，前出の入出力ポートから抜き出された同期情報の中から，２つの条件（ｃｃ１，ｃｃ２）を設定できる．これにより，（２）で述べたライン単位，フレーム単位の同期をとりながら比較演算を行う処理が可能となった．また，この同期情報を使用すると，プロセッサ間の同期を取ることができ，並列動作に有効である．

５．３．３　プロセッサとコントローラ間の通信機構

プロセッサは，上述のシーケンサにより自立的に動作するが，クラスタに付属するコントローラの制御を受け入れる機能がある．

画像処理においては，処理結果に応じて処理内容を変更したい場合があり，放送においては，演出効果を上げるために色変換を行った後に幾何学変換を行うといったように次々と処理を変えて行くのが普通である．しかも，この機能変更を画面上に妨害を与えないで行うことが要求される．

これを実現する機構は，５．２．５で述べたＲＥＳＴＡＲＴ命令とインタフェース・レジスタを用いて実現されている．このレジスタ２セット（Ａ，Ｂ）は，必ず一方がプロセッサ側に，もう一方がコントローラ側に向いている．いま，仮にＡセットがプロセッサに，Ｂセットがコントローラに向いていたとする．コントローラは，Ｂセットの０番地にスタート番地を，１～７番地にパラメータを与える．その後，垂直帰線期間になり，ＲＥＳＴＡＲＴ命令がデコードされると，シーケンサはレジスタのＡセットとＢセットを入れ替え，入れ替わったＢセットの０番地の内容をシーケンサのプログラムカウンタ（ＰＣ）にロードする．シーケンサは，この値をスタート番地として動作を開始する．パラメータが必要であれば，１～７番地から取り込む．
一方コントローラは，自分側に向いたＡセットにスタート番地等を書き込めば，次の処理を指示できる．この切り替えは，プロセッサが垂直ブランキング期間内にＲＥＳＴＡＲＴ命令を発しない限り行われないし，ＲＥＳＴＡＲＴ命令が発せられてもコントローラから次の処理の指示を与えていなければ行われない．従って，不用意に処理が中断して，画面が乱れることはない．インタフェース・レジスタの切り替えを利用すると，プロセッサがインタフェー

ス・レジスタにデータを書き込んでＲＥＳＴＡＲＴ命令を発すれば，コントローラにデータを返すこともできる．

また，リセット端子を用いてプロセッサをリセット状態にすると，コントローラからプログラムメモリとデータメモリの書き込みと読み出しを行うことができる．従って，ホストコンピュータは，ＬＡＮを通してコントローラと通信することにより，制御だけでなく，データとプログラムをプロセッサに送り込み，処理を行わせ，その結果を吸い上げることも可能である．

５．３．４　ネットワーク

ネットワークは，３２入力（プロセッサからの出力１６とクラスタ外からの入力１６）４８出力（プロセッサへの入力３２とクラスタ外への出力１６）のクロスバー・マトリックスであり，入力と出力間で任意の接続が可能である．１つの入力を同時に複数の出力に分配することも，複数の入力を時分割に１つの出力に集中することも可能であり，プロセッサをパイプライン状に接続することも，アレイ状に接続することも，さらに両者を組み合わせた形態をとることもできる．このネットワークは，１４．３ＭＨｚの速度で動作する．ネットワークの切り替えは，プロセッサと同様ネットワークに付属するシーケンサにより，マイクロプログラム制御される．ネットワークの各接続を制御するためには２４０ｂｉｔ（＝４８×５）幅のマイクロ命令が必要である．しかし，多くの画像処理では１フィールド内同し処理を行う場合か

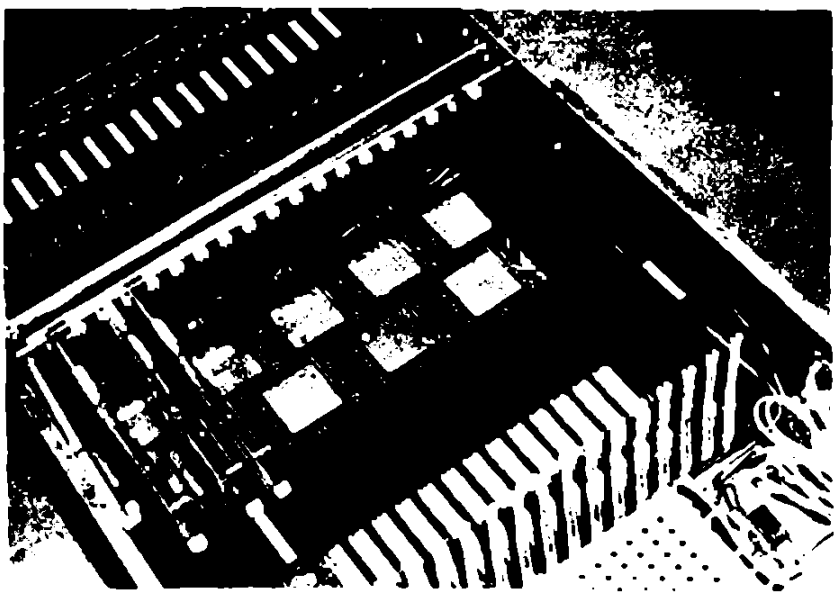


図５．１０　ネットワークの外観写真

多く，１フィールド内ほとんどのルートを固定しておいても問題無いので，１命令では，ひとつの出力をひとつの入力に接続するという制御を行うことにした．従って，全ての出力を確定するのに４８命令必要になる．しかし，一度セットされると，新たなる命令によって接続が変更されるまでルートは固定されているため問題はない．このように，フィールド単位を基本としているが，領域分割を用いたプロセッサの並列動作においては，画素単位やライン単位に切り換える必要がある．このために，出力１系統については，画素単位やライン単位の切り替え機能を付加した．マイクロプログラムメモリの容量は，１２８ｗ×１６ｂｉｔである．

ネットワークの主な部分は，ＣＭＯＳゲートアレイを用いてＬＳＩ化した．このＬＳＩについては，５．３．６で述べる．図５．１０にクラスタ内に実装されたネットワークの写真を示す．クラスタは，幅約４０ｃｍ×奥行き約６０ｃｍ×高さ約２０ｃｍの大きさで，プロセッサを挿入するマザーボード上に実装されている．

５．３．５　コントローラ

プロセッサとネットワークは，５．３．３及び５．３．４で述べたように，それぞれが有するシーケンサにより自立的に動作するが，コントローラ（ＭＣ６８０００）によっても制御できる．コントローラは，ホストコンピュータ，コントロールパネルから，ＬＡＮを通して送られてきたコマンド，データ，プログラムを解析して，プロセッサ及びネットワークに指示を与える．また，プロセッサの処理結果を解析して，プロセッサに新たなる指示を与えたり，ホストコンピュータにデータを送り返すなどの処理を行う．

５．３．６　プロセッサＬＳＩとネットワークＬＳＩ

プロセッサとネットワークの主な部分は，ＬＳＩ化されている．プロセッサに関しては，プログラムメモリ，データメモリ，ラインメモリを除いたほとんどの部分が，ネットワークに関しては，プログラムメモリも含めたほとんどの部分がＬＳＩ化されている[99]．プロセッサＬＳＩの構成図を図５．１１に，ネットワークＬＳＩの構成を図５．１２に示す．

プロセッサＬＳＩ（Ｐｉｃｏｔ）は，本システムのプロセッサとして使用することを主目的に設計されているが，単独のＤＳＰ（Digital Signal Processor）として，あるいは複数使用して各種映像機器に組み込んで使用することも可能である．データメモリに関しては，最大１６Ｍｗまで接続できるようにアドレス幅が２４ｂｉｔになっている．ＨＡＵとＶＡＵの各２２ｂｉｔの演算出力から，データメモリにアクセスするために，５．３．１で述べ

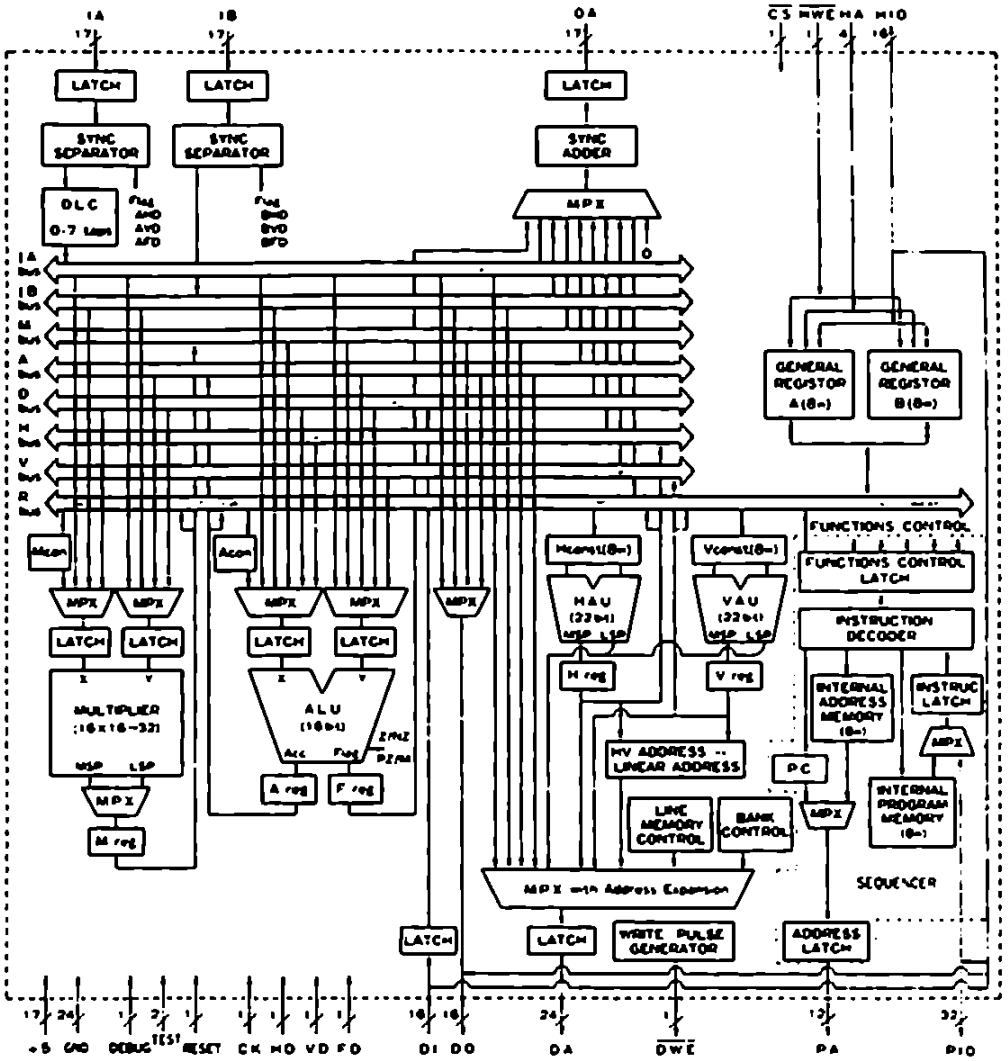


図5．11　プロセッサI.SIのブロック図

たHAU，VAUからそれぞれ10bit，8bitを抜き出して18bitにするモードに加え，HAU，VAUからそれぞれmビットを抜き出して連結し2mビット列とするアドレス変換回路を付加した．用意したモードは，512×512画素（m＝9），1024×1024画素（m＝10），2048×2048画素（m＝11），4096×4096画素（m＝12）の4つである．これにより，工業用によく用いられる512×512画素から，HDTVを超える大きな画像まで対応できる．また，データメモリには，RAMとROMを混在して使用できるため，特殊関数をテーブル化してROMとして搭載することも可能である．プログラムメモリも同様に，RAMだけでなくROMも使用可能であるので，ROM化して専用プロセッサとして

表5．3　プロセッサL.SIの主な仕様

Technology	1.5μm CMOS gate array
Chip size	14.95 mm × 14.95 mm
Number of devices	30,000 gates
Power supply	5 V
Power dissipation	1.0 W
Package	223-pin PGA
Clock frequency	14.3 MHz

表5．4　プロセッサＬＳＩの処理能力

Interframe differenciation	20ms(1Picot)
Noise reduction (2×2)	80ms(1Picot), 20ms(4Picots)
Convolution (3×3)	180ms(1Picot), 20ms(9Picots)
Binarization,Gray-scale conversion	20ms(1Picot)
Maximum detection (3×3)	180ms(1Picot), 20ms(9Picots)
Affin transformation	20ms(1Picot)
Ensemble between images	20ms(1Picot)

各種機器に組み込むことも可能である．

プロセッサL.SIは，１．５μｍＨＣ²ＭＯＳ２層配線のチャンネルレス型のゲートアレイを用いて試作した．回路規模が３０Ｋゲート程度となったが，複数バスを採用しているなどの要因により配線領域が大きくなることを考慮し，ｒａｗゲート数１３０Ｋのものを用いた．レイアウトは自動で行ったが，高速性を要するところは配線長が短くなるようにブロック毎に大まかに指定して行った．プロセッサＬＳＩの主な仕様を表５．３に，ＬＳＩの処理能力を表５．４に示す．

ネットワークＬＳＩ（Ｐｉｃｏｔ－ｎｅｔ）は，本システムのネットワークを小型化するために設計したものである．３２入力４８出力のネットワークであるが，ピン数の関係からデータ幅が１７ｂｉｔではなく，２ｂｉｔ幅

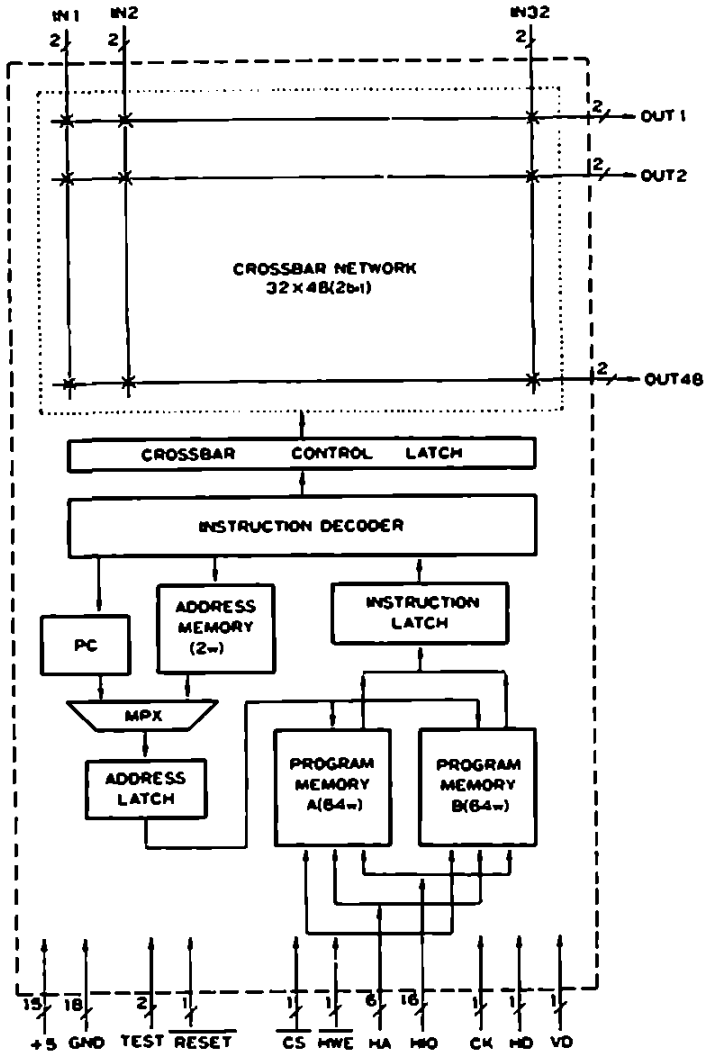


図5. 12 ネットワークLSIのブロック図

表5. 5 ネットワークLSIの主な仕様

Technology	1.5μm CMOS gate array
Chip size	14.95mm×14.95mm
Number of devices	28,500 gates
Power supply	5 V
Power dissipation	1.0 W
Package	223-pin PGA
Clock frequency	37.125MHz

である．このため，１クラスタに２つ１個使用する．ネットワークＬＳＩはプロセッサＬＳＩと同じデバイス技術を用いて実現した．ネットワークＬＳＩの主な仕様を表５．５に示す．プロセッサＬＳＩの仕様とほぼ同じであるが，動作周波数は３７．１２５ＭＨｚと高速である．ただし，Ｐｉｃｏｔ－ｓｙｓｔｅｍでは，プロセッサに合わせ１４．３ＭＨｚで動作させている．

この２種のＬＳＩは，Ｐｉｃｏｔ－ｓｙｓｔｅｍ以外にも，すでに使用されている．プロセッサＬＳＩは，東芝のＮＴＳＣ及びＨＤＴＶ用効果波形発生装置[7]に，ネットワークＬＳＩは，ＮＨＫ及びＪＳＢの映像音声分配切替システム[7][8]に組み込まれて使用されている．

５．４　ソフトウェア構成

５．４．１　処理の流れ

Ｐｉｃｏｔ－ｓｙｓｔｅｍの処理の流れを，アフィン変換を例に取り上げて説明する．図５．１３に，処理系統を示す．アフィン変換は，（２．１）（２．２）式で実現できる．（２．１）式の演算をＨＡＵで，（２．２）式の演算をＶＡＵで行うことができるので，静止画であれば１プロセッサでできる．しかし，動画像を通す必要があるため，２プロセッサを使用する．すなわち，プロセッサのデータメモリのサイクルタイムは，１サイクル（７０ｎｓ）なので，書き込みと読み出しを同時にできない．そこで，２台のプロセッサをインターリーブして並列動作させる．プロセッサ１は，第１フィールドでメモリに書き込み，第２フィールドで読み出す．プロセッサ２は，その逆で第２フィールドで書き込み，第１フィールドで読み出す．書き込み時は，アドレスを変換しないで書き込み，読み出し時にアフィン変換したアドレスをアドレス回路で発生させてメモリを読み出す．プロセッサ３と４は，プロセッサ１と２で１フィールド遅れとなった画像をさらに１フィールド遅

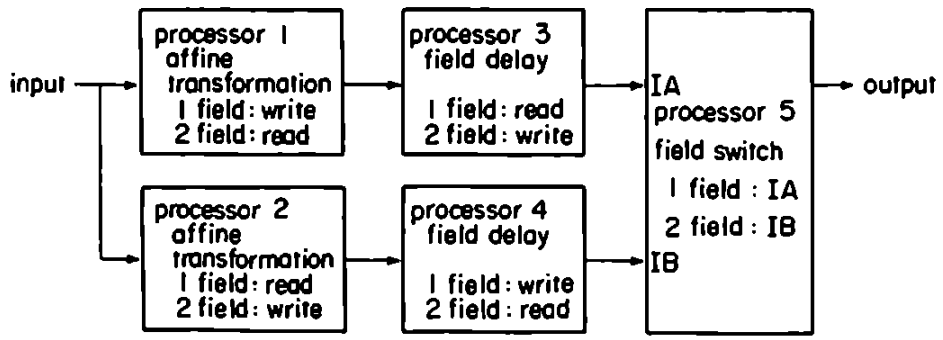


図５．１３　アフィン変換の信号の流れ

らせることでインタレース走査している映像の位置を正規にするためのものである．プロセッサ５は，フィールド毎に書き込み読み出しが切り換えられるプロセッサ３と４の中から，読み出しモードになっているプロセッサを選択して出力する処理を行う．以上は，モノクロ画像の場合であるので，ＲＧＢのカラー画像の場合は，３倍の１５プロセッサ必要である．

また，コントロールパネル上のジョイスティックの値を，水平拡大率，垂直拡大率，回転角，水平移動量，垂直移動量に割り当て，コントローラではこれらの値から（２．１）（２．２）式のＡ～Ｄのパラメータを算出し，プロセッサ１とプロセッサ２のインタフェース・レジスタに渡すようにする．これにより，コントロールパネルのジョイスティックを動かすと，リアルタイムで拡大・縮小，回転，移動を行うことができる．

５．４．２　ソフトウェア開発環境

Ｐｉｃｏｔ－ｓｙｓｔｅｍを効率的に動作させるために構築したソフトウェア環境について述べる．ソフトウェア開発は，ホストコンピュータを中心に行う．ソフトウェアは，開発系，実行系，デバッグ系に大別できる．以下，これらについて順に述べる．

（１）開発系

プログラムを開発するためのソフトウェア群である．プログラム開発は，図５．１４の手順に従って行う．まず，プロセッサ，ネットワーク，コントローラの３種のプログラムを作り，これをアセンブル，リンクし，パケット化する．更にこれをＬＡＮを通してダウンロードする．プロセッサとネットワークは，アーキテクチャに適合した専用の言語を開発した．コントローラは，６８０００用のＣ言語を用いる．また，プロセッサ内のデータメモリにロードするルックアップテーブルを作成するツールも用意した．

プロセッサに関しては，映像信号処理をかなり意識したプロセッサの制御構造を簡易に表現できる図５．１５に示すような専用の言語（言語仕様は，付録１参照）でプログラム開発を行う．図５．１５は，画面を左右に分割して，画面左側にＩＡ入力を右側にＩＢ入力をワイプして表示するためのプログラム例である．’以降はコメントである．分割位置の判定は，ＨＡＵの演算フラッグhauf2 を用いて行う．画面左からカウントを開始しhcmp2 を超えてhauf2 が立ったら，出力を入力ＩＡからＩＢに切り替える．①は，プロセッサ番号の宣言文である．②は，コントローラがこの関数をコールするために使うモジュール名である．③～⑥は，有効画面部分が始まる前の垂直帰線期間に行う初期値の設定などである．⑦は，垂直帰線期間が終了するのを待っている．⑧～⑯は，有効画面中の処理である．⑰は，一連の処理が終了し，

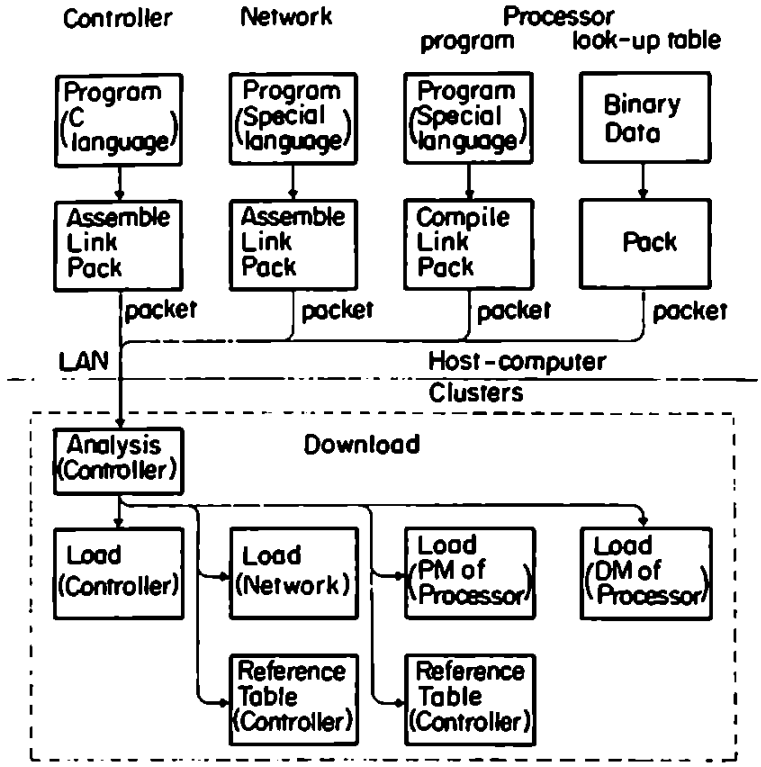


図５．１４　開発系ソフトウェア

```
picot4[                          '①processor number
"sample1";                       '②module name
    dlc(0);                      '③set delay line compensator
    ldr(hint1, 0);               '④set hint1(initial value of hau)
    ldr(hinc1, 1.0);             '⑤set hinc1(increment value of hau)
    mgs(greg1, hcmp2);           '⑥set hcmp1(comparison value of hau)
    while(svd) out(clr);         '⑦wait for active image area
loop:                            '⑧process for active image area
    while(shd) au(hint1/reg1);   '⑨set hau hint1
    au(hau/reg1, hinc1);         '⑩start incrementation of hau
    while(!shd) {                '⑪no code generated
      while(!hauf2) out(la);     '⑫repeat/cont(hauf2)/jmp⑮(shd)
      while( hauf2) out(lb);     '⑬repeat/jmp⑫(!hauf2)/cont(shd)
    }                            '⑭no code generated
    if (!svd) goto loop;         '⑮jump ⑨ if !svd
    return;                      '⑯control returns to controller
]                                '⑰end of program for Picot4
```

図５．１５　プロセッサのプログラム例

制御の主導権をコントローラに返す命令である．これにより，次の処理をコントローラから受けることができる．フィールドをまたがって処理したい場合は，ｒｅｔｕｒｎ文を書かなければ，コントローラに制御が戻ることはない．

ここで，同期信号を用いたシステム制御と一度に２つのフラッグをチェックできる機能を明示的にかつ分かりやすく記述するために設けたｗｈｉｌｅ文について述べる．図５．１５の⑪～⑭に示す書式で記述する．外側のネスト（入れ子）の条件が優先順位が高い条件１（ｃｃ１）になり，内側の条件が条件２（ｃｃ２）になる．⑪⑭は，マイクロコードは生成されない．⑫⑬はｃｃ１，ｃｃ２を設定する２条件３分岐のコードが生成される．⑫は，ｃｃ１にシステムＨＤが，ｃｃ２にhauf2 がセットされる．ＨＤが来たとき⑮にジャンプし，ＨＡＵの演算フラッグであるhauf2 が立つと次の行⑬に移る．それ以外の場合は，⑫を繰り返す．一方⑬は，ｃｃ１にシステムＨＤが，ｃｃ２に!hauf2（！はｎｏｔを示す）がセットされる．ＨＤが来たとき次の行⑮に移り，hauf2が立たないとき⑫にジャンプする．それ以外は，⑬を繰り返す．ネストは，ハードウェアの制約から最大２であるので，それ以上設定しようとするとアセンブル時にエラーとなる．値hcmp2 が８９４以上の場合，一般にはＨＤを読み飛ばしてしまうおそれがあるが，ここでは優先度の高いｃｃ１にＨＤがセットされているために，hcmp2 の値が依存せず⑮に飛ぶことができ，プロセッサの特徴である実時間処理の機能を簡易に表現できる．shd （システムのＨＤ）やsvd（システムのＶＤ）の代わりにahd（入力ＩＡのＨＤ），avd （入力ＩＡのＶＤ）をセットすると，システムではなく画像にタグとしてつけられた同期情報を使ってデータ駆動的に動作できる．

また，５．３．２で述べた実時間性を維持するための機構ＩＡＭとＩＰＭに関しては，ＩＡＭについてはジャンプ先のアドレス＋１を，ＩＰＭについてはジャンプ先のプログラムを全ての処理に先だちロードしておく必要がある．このためのマイクロコードを自動的に生成して挿入する言語処理系とし，プログラマの負担を抑えた．

ネットワーク用ソフトウェアは，プロセッサと類似した図５．１６に示す制御構造の言語（言語仕様は，付録２参照）で記述する．図５．１６中のｗｈｉｌｅ文では水平同期ＨＤの期間以外でプロセッサ２と３の出力を画素単位に切り替える例である．

以上のプログラムは，アセンブル，リンクされ，さらにＬＡＮ上に送出するためにパケット化される．パケットには，クラスタ番号，プロセッサ番号，モジュール名などの識別情報も含まれる．これらの情報により，プログラムのダウンロードを行う．

```
network[                    'program for network
"sample2";                  'module name
    net(in1 , pu1a);        'input1 → processor1's IA
    net(in2 , pu1b);        'input2 → processor1's IB
    net(pu1, pu2a);         'processor1's OA → processor2's IA
    net(pu2, pu3a);         'processor2's OA → processor3's IA
    while(!shd) {           'switch pixel by pixel
      net(pu2, out2);       'processor2's OA → output2
      net(pu3, out2);       'processor3's OA → output2
    }                       '
    return                  '
]                           '
```

図５．１６　ネットワークのプログラム例

コントローラでは，コントロールパネルの値を解析して，プロセッサやネットワークを制御する．プログラムはＣ言語で開発する．詳しくは，次で述べる．

（２）実行系

実行系は，クラスタ群を動作させて，動画像処理を実際に行うためのソフトウェア群である．人間系とのマッチングをとるためには，処理の応答時間は１～２フィールド（１６～３３ｍ秒）以内が望ましいので，実時間性を重視した設計方針を取った．ＯＳ（Operating System）のオーバーヘッドが大きい汎用のパーソナルコンピュータでは，実時間性に問題があるため，ホストコンピュータでは，処理の開始，停止だけを行い，実行時は専用のリアルタイムＯＳを搭載したコントローラが主体となる方式にした．

図５．１７が実行時のシーケンスである．最初に，ホストコンピュータからスタートをコントローラに指示する（図５．１７中❶）．コントローラは，ＶＤに同期してコントロールパネルにパケットの送信要求を出す（図５．１７中①）．コントロールパネルは卓面の状態をパケット化して出力する（図５．１７中②）．データは，加工されることなくジョイスティックなどの値そのものが出力される．コントローラは，パケットを受信（図５．１７中③）．解析（図５．１７中④）し，データ処理して（図５．１７中⑤），プロセッサとネットワークにパラメータを与える（図５．１７中⑥）．プロセッサとネットワークは，パラメータを受け取り，次のＶＤがきたらプログラムを切り替え（図５．１７中⑦），処理を開始する（図５．１７中⑧）．以降①

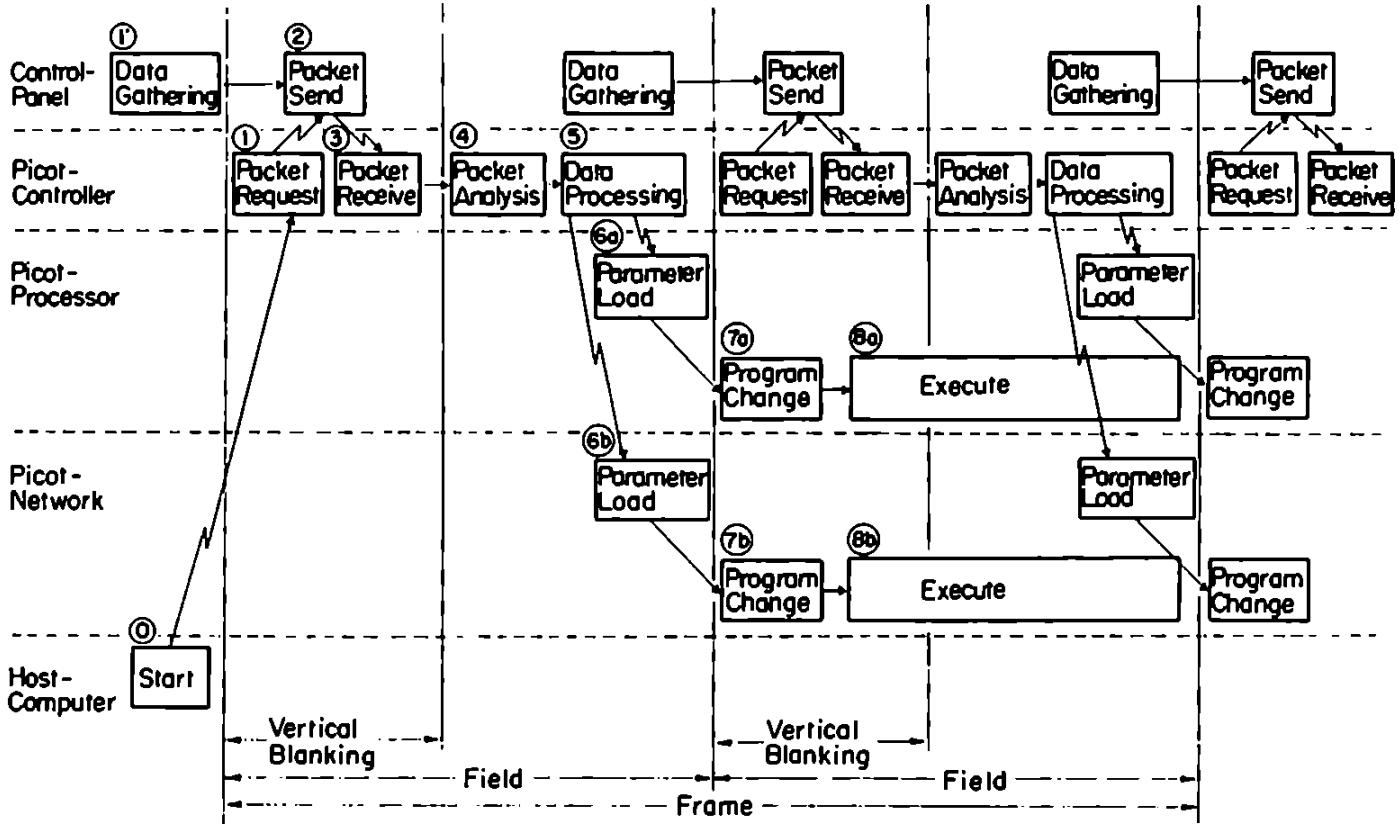


図５．１７　実行時の制御の流れ

～⑧の処理を繰り返す.

このうち，プログラマは，パケット解析とデータ処理の部分（図5．17中④⑤）をプログラムするだけでよい．その他の処理は，コントローラのOSが行う．また，コントローラのOSは，プログラムのダウンロード時にパケットに付属した識別情報に基づいてモジュール名と絶対番地の対応表を作成するようにしている．これにより，コントローラのCプログラムでは，絶対番地ではなくモジュール名でプロセッサやネットワークのプログラムを関数を呼ぶようにコールできる．前もって，ネットワークとプロセッサのプログラムメモリに，平滑化，微分などの基本ルーチンをライブラリ化してロードしておけば，これをコールするコントローラのCプログラムを作成するだけでも，1連の画像処理を実現できる.

（3）デバッグ系

デバッグ系は，ソフトウェア開発を支援するためのソフトウェア群である．プロセッサのレジスタ，データメモリの内容，コントローラの状態，コントローラパネルから受信したパケットの履歴をホストコンピュータに引き上げることができ，ＩＣＥ（In Circuit Emulator）等を使用しなくても，システムの状態をチェックできる．これによりデバッグ作業の効率化が計れた.

5．4．3　ＰｉｃＰＥｎ

Ｐｉｃｏｔ－ｓｙｓｔｅｍのようなマルチプロセッサでは，ソフトウェア開発が複雑になる．5．4．2で述べた開発環境では，実現したい大きな処理を各プロセッサがビデオレートで処理可能な小さなタスクに分割し，プロセッサに割り当てるのは，プログラマの負担である．このタスク分割を行うためには，システムの内部構成について細かな知識が必要である．アプリケーション・ソフトウェアを増やすためには，システムのハードウェアの知識を持たなくても，簡単にプログラムが作成できるツールが必要である．このようなツールとして，ＰｉｃＰＥｎ（Picot Programming Environment）を開発した[74]．ＰｉｃＰＥｎでは，実現したい処理をホストコンピュータ上で図5．18に示すようにシグナルフローグラフの形で記述すれば，自動的に前述のプロセッサ及びネットワーク用マクロアセンブラ，コントローラ用Ｃプログラムを生成するツールである．アイコン化されたモジュールは，差分，混合，フレームディレイなどの処理を行うものである．モジュールは，階層化されており，よりプリミティブなモジュールで記述されている．従って，モジュールは，プロセッサと1対1に対応しているのではなく，機能を記述している．モジュールを分解あるいは統合し，タスクをプロセッサに割り当てる作業は，ＰｉｃＰＥｎが自動的に行う．この際，

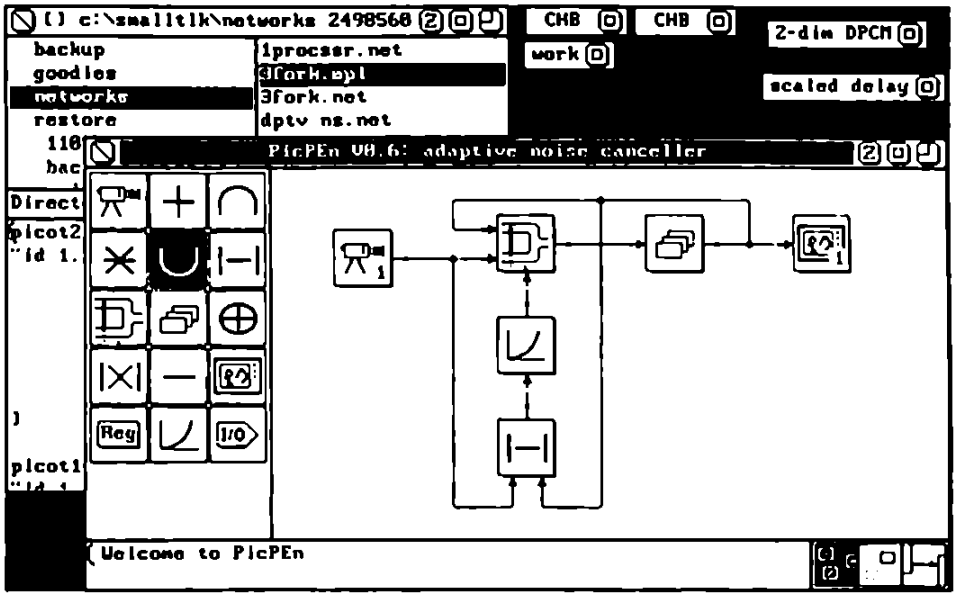


図5．１８　ＰｉｃＰＥｎの表示画面

（１）プロセッサは２入力，１出力である．

（２）プロセッサには，乗算器，ＡＬＵが各１個ある．

（３）プロセッサでは，フレームメモリとルックアップテーブルの共用はできない．

などの制約条件を考慮しつつ，自動的にタスクをプロセッサに割り当てる．また，Ｐｉｃｏｔ－ｓｙｓｔｅｍでは，パイプライン化のために多くのディレイが生じ，異なるパスを通った時の遅延差が問題になるが，ＰｉｃＰＥｎがディレイ量を計算して必要に応じて自動的にディレイを挿入し，遅延差を補償する[7]．これらの処理は，クラスタをまたがっても解析可能になっている．

ＰｉｃＰＥｎは，ホストコンピュータ上のＳｍａｌｌｔａｌｋ上にインプリメントした．それは，Ｓｍａｌｌｔａｌｋが高度なグラフィック機能を有することや，オブジェクトの概念が階層化したモジュール構造などにマッチしているためである．

５．５　各種映像信号処理の適用と評価

Ｐｉｃｏｔ－ｓｙｓｔｅｍでは，入出力される画像信号の標本化周期とプロセッサの命令サイクルが同じであるため，１画素当り１ステップの命令の実行を基本としている．しかし，プロセッサの並列・パイプライン動作により，プロセッサ数に比例したステップ数に相当する処理を実行でき，複雑な

表5．6　各種映像信号処理のＰｉｃｏｔ－ｓｙｓｔｅｍへの適用

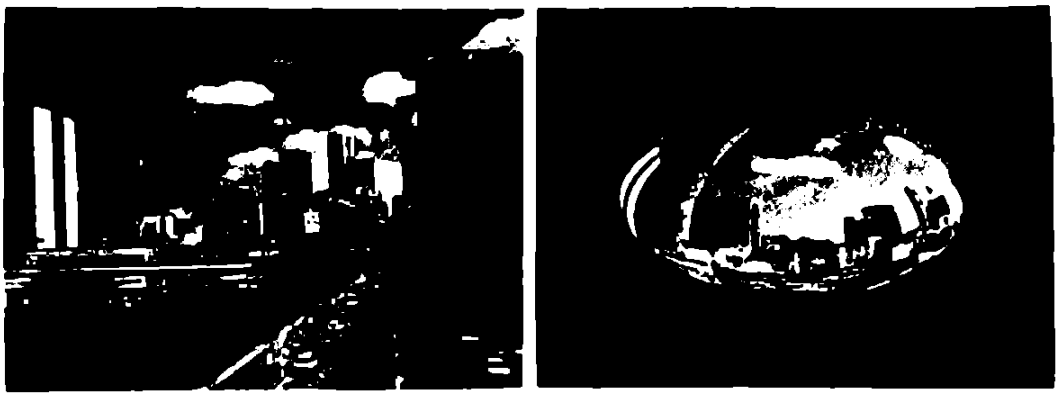
処理内容		必要プロセッサ数*)
点演算	濃度値変換	3
	閾値処理	3
近傍演算	ラプラシャン（３×３）	9
	スムージング（２×２）	4
幾何学変換	アフィン変換	１５
	テクスチャマッピング	３３
色処理	色補正	3
	クロマキー合成	１８
フレーム間処理	フレーム間差分	２１
	動物体抽出	１８
映像間処理	映像混合	6
	ワイプ処理	3

*)カラー映像信号（RS170A準拠，RGB，525本/60Hz，インターレース）の場合

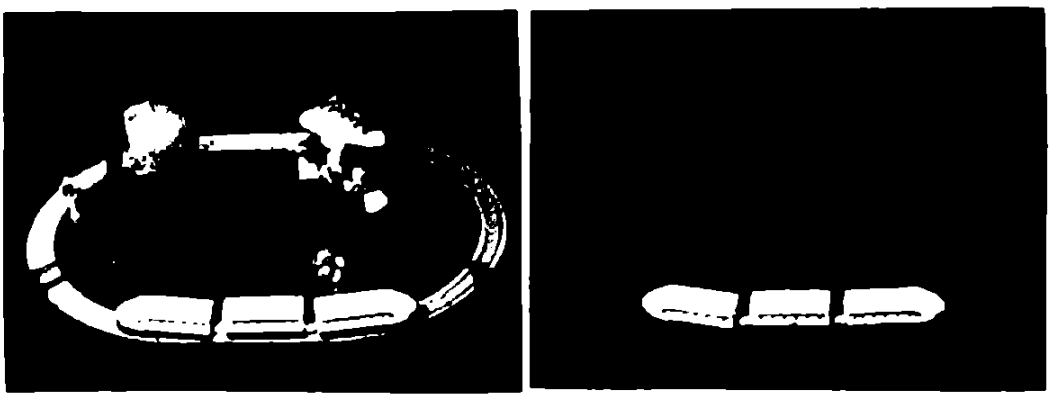
表5．7　プロセッサの処理能力

model(1)	SUN		NEWS	
processing(2)	elaps	ratio	elaps	ratio
addition, subtraction	0.27sec	8.1	0.49sec	15.0
maximum, minimum	0.24sec	7.2	0.63sec	19.2
absolute	0.41sec	12.6	0.76sec	23.4
multiplication(mpy)	0.88sec	27.0	1.43sec	43.8
multiplication & addition	0.91sec	27.0	1.62sec	43.8
read or write frame-memory	0.35sec	10.8	1.84sec	56.4
interframe difference	0.39sec	12.0	0.62sec	18.9
affine transformation	1.57sec	48.0	3.07sec	90.9
affine & mpy & addition	2.34sec	71.7	4.35sec	133.2
convolution(3*3)(3)	7.35sec	25.0	13.13sec	44.6

（注１）SUN ：Sun-3/260（6649Dhrystones/sec, 3.8MIPS）
NEWS：NWS-830　（4242Dhrystones/sec, 2.4MIPS）
VAX11/780（1757Dhrystones/sec）を1MIPSとして換算
（注２）Picot-processor（フレームメモリ付）１台で可能な処理を想定し，白黒画像（754*483画素）を対象にした．
（注３）convolutionには，Picot-processor９台必要なので，ratio（対Picot比）は，1/9倍して算出した．



（ａ）テクスチャ・マッピング



（ｂ）動物体抽出

図５．１９　処理例

処理もビデオレート処理できる．従って，処理の複雑度に依存して必要なプロセッサ数が決定される．表５．６に，主な映像信号処理を行った場合に必要なプロセッサ数を示す．処理例を，図５．１９に示す．（ａ）は，球体の表面へのテクスチャマッピング，（ｂ）は，動物体抽出の例である．

Ｐｉｃｏｔ－ｓｙｓｔｅｍの処理能力を評価するために，Ｐｉｃｏｔ－ｓｙｓｔｅｍでビデオレート処理できる処理を，通常のコンピュータで処理した場合の処理時間を計測することにより比較した．表５．７に処理結果を示す．elaps は処理に要した時間であり，ratio は処理時間の対Ｐｉｃｏｔプロセッサ比である．実験では，標準ＴＶ方式の１４．３ＭＨｚで標本化した画像，すなわち有効画素数が横７５４画素×縦４８３画素の白黒画像を対象とした．また，処理内容は，１フレーム分のデータメモリを有するプロセッサ１台で，ビデオレート（１／３０秒以内）で処理可能なものを主対象とした．評価したコンピュータは，Ｓｕｎ－３／２６０とＮＥＷＳ８３０

である．プログラム作成に当たっては，レジスタ変数，ポインタを使うなどして，極力高速化を図った．表５．７によれば，画像処理の内容により違いはあるが，Ｓｕｎ（３．８ＭＩＰＳ）の最大７２倍，ＮＥＷＳ（２．４ＭＩＰＳ）の最大１３３倍の高速処理ができる．このことから１台のプロセッサは，最大３００ＭＩＰＳ程度の汎用コンピュータに相当する画像処理能力を発揮できることになる．

上述のプロセッサの処理能力から計算すると，１６台のプロセッサを有するクラスタは，１クラスタ当り４，８００ＭＩＰＳ，４クラスタでは１９，２００ＭＩＰＳ，１２クラスタでは５７，６００ＭＩＰＳの処理能力になる．しかし，プロセッサの全ての演算要素をいつも使うとは限らないので，これらの値は最大値ととらえるべきであろう．アフィン変換を例に取り上げる．表５．６によれば，アフィン変換はプロセッサ１５台で実現できる．処理内容の内訳は，表５．７のaffine transformation の処理６台，read or write memoryの処理６台，addition３台であるので，ratio を加重平均して求めると，１５台をフルに使ったばあいの約１／３の能力の１５２５ＭＩＰＳに相当する能力を使用している．

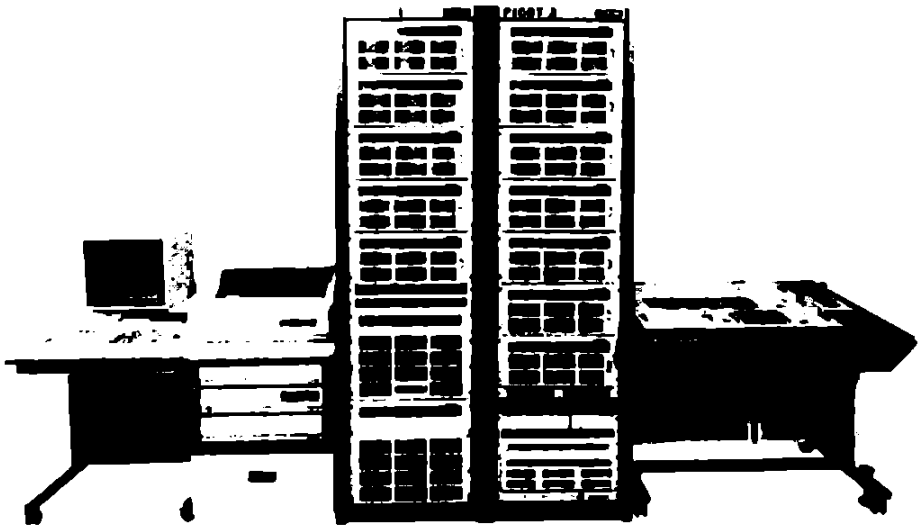
５．６　適用例

５．６．１　ビデオスイッチャ

プログラム可能な映像機器の番組制作現場への導入効果を検証する目的で，Ｐｉｃｏｔ－ｓｙｓｔｅｍをベースにして，ビデオスイッチャを実用化した．現在，番組制作用の設備としてＮＨＫ放送センタで試験運用している[7][8]．１２クラスタ構成のシステムである．図５．２０に外観写真を，表５．８に主な仕様を示す．

本システムは，カラーコレクタ，３列の効果列（Mix/Effect）などかなり大規模のビデオスイッチャに相当する機能を全てソフトウェアで実現している．図５．２１に従来形のビデオスイッチャによる構成法に置き換えた場合の構成図を示す．ディゾルブ，ワイプ，クロマキー，拡大縮小，デフォーカス，エッジスーパーなどの一般的な機能に加え，マット合成機能，ウェーブ効果などの新しい機能もプログラムした[7][8]．

番組制作においては，幾何学変換などの特殊効果が多用されている．このため，システムの処理能力を上げるために，幾何学変換時に生じる折り返し除去用のフィルタボードを開発した[7][8]．このフィルタボードは，水平２４タップ，垂直９タップのＦＩＲ(Finite Impulse Response) フィルタである．プロセッサと同一形状のボードとし，置き換えられるようにした．本スイッチャでは，各クラスタに１台挿入している．



左から，ホストコンピュータ，本体(A/D，D/A，クラスタ)，操作卓

図5．２０　ビデオスイッチャの外観写真

表5．８　ビデオスイッチャの主な仕様

映像入力	アナログコンポジット(ＮＴＳＣ)８系統 標本化数１４．３MHz，量子化数　９bit
映像出力	アナログコンポジット(ＮＴＳＣ)３系統　(ON-AIR,NEXT,MONITOR) 標本化数１４．３MHz，量子化数１０bit
内部信号形式	コンポーネント（Y，Ｒ－Ｙ，Ｂ－Ｙ） Ｙ　：１４．３MHz，１６bit(2's Complement) R-Y,B-Y：７．１５MHz，１６bit(2's Complement)　時間軸多重化
処理速度	７０ｎｓ/ｐｉｘｅｌ，３３ｍｓ/ｆｒａｍｅ
クラスタ数	１２
プロセッサ数	１９６（うち１６台は，フィルタ処理専用）
機能	ColorCorrector（色補正） ゲイン，ペデスタル，ガンマ，ヒュー，サチュレーションなど Mix/Effect１（効果列１） ディゾルブ，ＮＡＭ，マット合成 ワイプ(四角，円，菱形，星，ランダム，ポジション，高調波，変調波など) 特殊効果(拡大・縮小，デフォーカス，モザイク，ウェーブなど) Mix/Effect２（効果列２） ディゾルブ，ＮＡＭ，ワイプ，クロマキー DownStream（ダウンストリーム） ディゾルブ，ＮＡＭ，ワイプ，エッジスーパー，プレビュー，モニタ機能

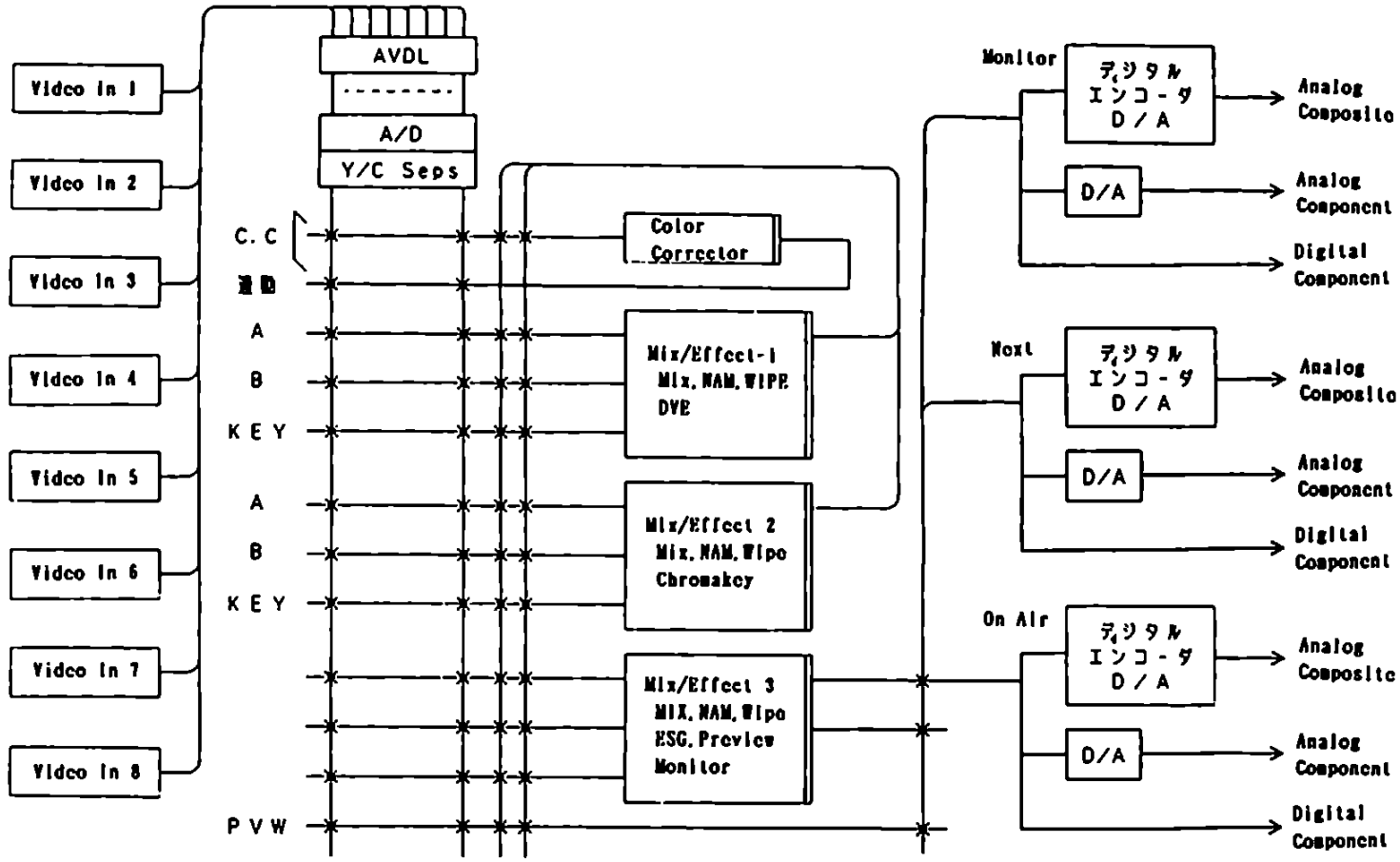


図5. 21　従来のビデオスイッチャに置き換えた場合の構成図

本システムは，ホストコンピュータ上のキーボードだけからでもコントロール可能であるが，制作スタジオでの日々の運用を考慮し，専用の操作卓（図５．２０右側）を用意した．従来型のビデオスイッチャの操作卓の操作性（マン・マシン・インタフェース）を継承すると共に，Ｐｉｃｏｔ－ｓｙｓｔｅｍのもつ柔軟性の両者を整合するように作られている．大型ＥＬパネルを使用しており，メニュー画面などもソフトウェアにより変更可能で拡張性に優れている．

本システムは，機能が固定の専用機器を組み合わせて作られていた従来の番組制作システムとは大きく異なるもので，導入後も新機能の追加が容易である．現在も，現場の番組制作技術者により，新しい効果が作成されており，拡張性，柔軟性が確認された[11]．

５．６．２　特殊効果

演出効果を上げるために，シーンチェンジなどでは特殊効果と呼ばれる映像効果が頻繁に使われる．特殊効果は，次々と新しい効果が求められる一方で飽きらやすい効果でもある．これまでにも，様々な特殊効果機器が開発されてきたが，今では見向きもされない効果機器も少なくない．ハードウェアではなくソフトウェアで効果を実現できるＰｉｃｏｔ－ｓｙｓｔｅｍは，このような一過性の強い特殊効果などの用途に適したマシンであり，ハードウェアが陳腐化することがない．

Ｐｉｃｏｔ－ｓｙｓｔｅｍで実現した特殊効果の１例を，図５．２２に示す．(a)は前景の青色部分を抜き取り別の背景を合成した効果，(b)は２つの映像を波状的に切り換えてゆく効果の途中経過，(c)は正弦波を用いてアドレスを変調してマッピングした効果，(d)は乱数を使ってアドレスを変調しシーンチェンジに使った効果，(e)は非直線位相の残像効果，(f)は背景に縮小画像をはめ込んだ効果である．(a)～(d)は２クラスタ，(e)は３クラスタ，(f)は４クラスタ使用した．

５．６．３　画質補正

画質補正処理として，①ノイズ除去，②色補正，③フリッカ妨害軽減などをＰｉｃｏｔ－ｓｙｓｔｅｍ上で試みた．

①の例では，信号成分のフレーム間相関を利用したタイプのノイズリデューサを実現した．動きの検出精度を上げるために，フレーム間差分を水平方向にフィルタをかけて雑音成分と動き成分の識別を行う手法を用いた[12]．

②の例では，ゲイン，ペデスタル，ガンマ，ルミナンス，サチュレーション，ヒューを変更できるカラーコレクタを実現した．このうち，非線形演算であ

（a）クロマキー

（b）ワイプ

（c）ウェーブ

（d）ハースト

（e）残像

（f）DVE

図5．22　効果例

るガンマに関しては，２５６種類のガンマ・テーブルを１ｋｗ単位でデータメモリ（２５６ｋｗ）にロードして使っており，γ＝０．５～２．０まで可変にできる．ゲインについては０～１５０％，ペデスタルについては±５０％までと，通常のカラーコレクタに比べ可変範囲を大きくした．入出力がＹ：Ｒ－Ｙ：Ｂ－Ｙ（４：２：２）信号に対し，１クラスタで実現できた．

③の例では，１９８８年のソウル・オリンピック放送において，生じたフリッカ妨害を軽減させた．オリンピック放送においては，限られた映像伝送回線を最大限に使用するために，１つの回線に２つの映像を多重化して伝送する２画面伝送装置が通常使用されている[4]．ソウル・オリンピック放送で使用された２画面伝送装置[5][6]においては，送信側でフィールド・オフセット・サブサンプリングにより，斜め方向の解像度を落として，２つの映像を多重化する方式を採用していた．ところが，伝送回線の特性不良から，受信側で正しくリサンプルできず，サンプル位相が崩れたために，縦方向のエッジ部分に特に大きなフリッカ妨害を生じた．妨害の発生が発覚したのがオリンピック開催の直前のために改良を加える時間的余裕が無く，また既存の機器でも取り除くこともできなかった．

そこで，Ｐｉｃｏｔ－ｓｙｓｔｅｍを使用して妨害の軽減を試みた．現象を検討したところ，フィールド間で高域成分を除去すれば有効であることがわかり，図５．２３に示す系統で除去することにした．処理は，コンポジットのままで行い，１クラスタで実現することができた．フィールド間のフリッカ妨害のために静止画像にすると効果が分かりにくいが，図５．２４に処理前と処理後の画像を示す．現象の検討からプログラム作成，デバックまで３日程度で完成した．ソフトウェアによる映像信号処理の有効性が放送現場で確認される結果となった．

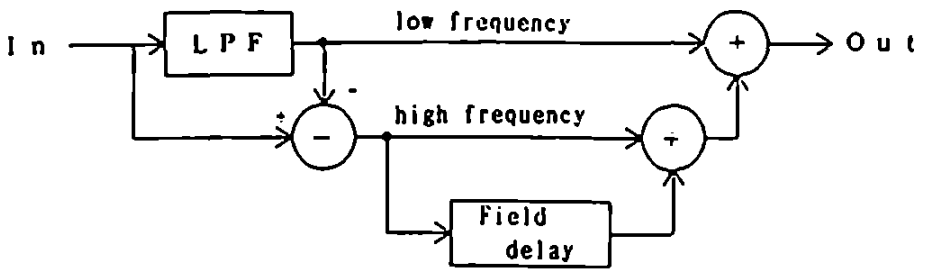
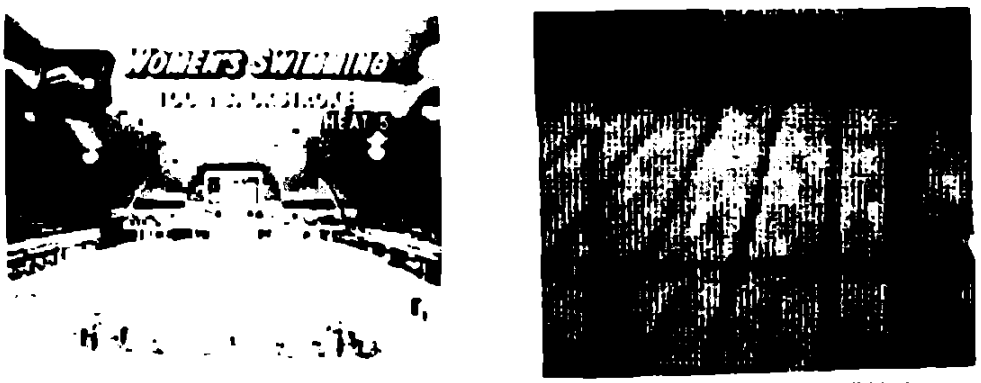


図５．２３　フリッカ軽減の系統図

（ａ）処理前



（ｂ）処理後

図５．２４　フリッカ軽減例

５．６．４　動き検出

動き検出は，方式変換装置，ディジタル伝送装置などではよく使われているが，番組制作機器ではこれまであまり使われていない．画像処理的な手法を番組制作に導入すると新しい映像表現が可能となる．その１例として，動き検出を用いた映像効果を試みた．

２．４．３で述べた動き検出法のひとつであるマッチング法をＰｉｃｏｌ－ｓｙｓｔｅｍにインプリメントした場合，（２．３６）式を計算するのに要する時間（フィールド数）は，

$$N_t = \frac{(S_m \times N_s + S_s) \times N_s + S_s + S_o}{f \times f_t} \qquad (5.\ 2)$$

である．ここで，

S_m：１点のマッチングに要するステップ数（＝14）

S_s：マッチングに付随して行われる処理に要するステップ数（＝66）

S_i：初期化などに要するステップ数（＝22）

S_e：動ベクトル決定などに要するステップ数（＝36）

N_a：マッチング対象の画素数

N_s：検索対象となる範囲（画素数）

f_s：演算速度すなわち標本化周波数（＝14.3MHz）

f_v：フィールド周期（＝1/60sec）

である．これを，グラフで示すと図５．２５になる．図５．２５によれば，検索範囲を水平方向－１６～＋１５画素，垂直方向－８～＋７ラインとし，マッチング点数を６０画素とした場合，２フィールドでマッチング処理が完了する．従って，動ベクトル検出は，メモリへの書き込みに１フィールド，マッチング処理に２フィールド，マッチング結果に基づいてパラメータを変更する処理に１フィールドの合計４フィールド必要である．これをビデオレートで実現するためには，図５．２６に示すように４プロセッサを時間をずらして並列動作させる．図５．２７にこの方式で行った動き検出の例を示す．図５．２７（ａ）は，顔を自動的に追跡して＋印で表示した例である．最初に顔の部分を指定して，その後はマッチング法により自動的に追跡した．＋印の代わりにモザイクを用いた例が，図５．２７（ｂ）である．従来より人の顔などをぼかすためにモザイクなどを用いているが，この動き検出手法を用いると最初に指定するだけでその後は自動で追跡しモザイクをかけること

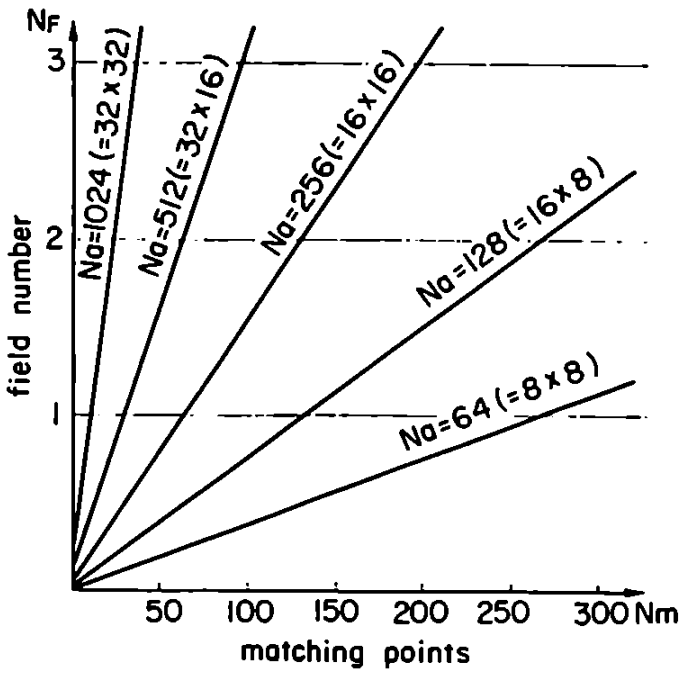


図５．２５　所要フィールド数

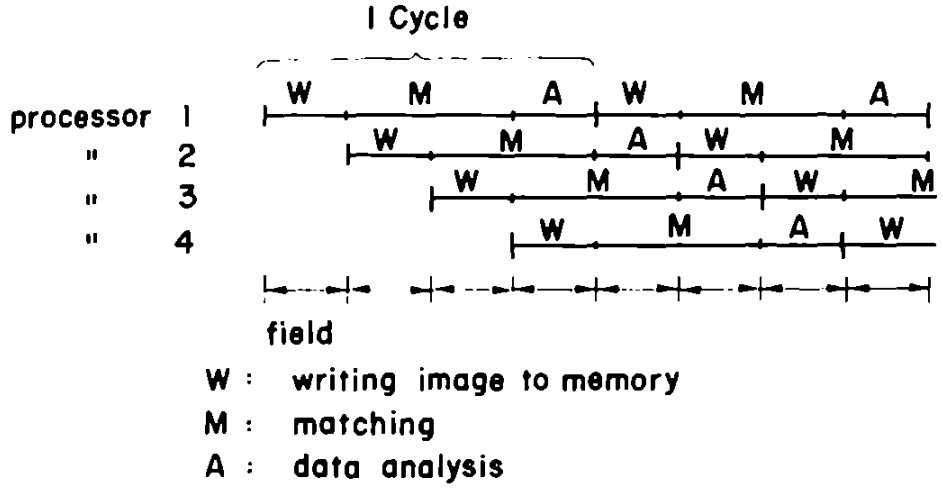


図5．２６　４プロセッサによる並列処理

（ａ）＋印表示例　　（ｂ）モザイク表示例

図5．２７　動き追跡例

かできる．

このＰｉｃｏｔ－ｓｙｓｔｅｍによる動き検出は，コンピュータグラフィックスで生成したキャラクタのリアルタイム制御「やマラソンランナーのピッチ計測」にも使われた．

5．6．5　映像合成

２．４で述べた映像合成法をＰｉｃｏｔ－ｓｙｓｔｅｍｉでビデオレートで処理させることを試みた．

（１）大きさ・形を変えず平行移動する場合

２．４．４のシミュレーション結果に基づき，各エリア１５画素の４エリ

アを設定して，撮影映像を用いて映像合成実験を行った．実験では，検索範囲を水平方向－１６～＋１５画素，垂直方向－８～＋７ラインとした．この場合は，図５．２５より２フィールドでマッチング処理が完了することがわかる．従って，動ベクトル検出は，メモリへの書き込みに１フィールド，マッチング処理に２フィールド，マッチング結果に基づいてキーを変形する処理に１フィールドの合計４フィールド必要になるため，４プロセッサを割り当てればよい．動ベクトルは４フィールド遅れで検出されるので，映像合成のためには原画像を４フィールド遅らせる処理が必要である．この処理に加え，キー作成（入力），検出点の指定の処理などを含めて合計３クラスタ使用した．撮影映像を用いて映像合成実験を行ったところ，シミュレーションと同程度，すなわち±２画素未満の検出精度の合成映像が得られた．

（２）回転・ズームを伴う場合

２．４．４のシミュレーション結果によれば，３２ヶ所以上の検出エリア（各エリア１６×１６画素）を設定し，最小二乗法により変換パラメータを求める必要がある．ところが，この処理をそのままＰｉｃｏｔ－ｓｙｓｔｅｍ上にインプリメントしても，プロセッサ台数に比べ演算量が多くビデオレート処理することは難しい．このため，回転を除いた拡大・縮小，移動する動きに限定して実時間動作を検証することにした．回転を除くと，（２．３７）（２．３８）式は，

$$X = a \cdot x + b \qquad (5.\ 3)$$

$$Y = a \cdot y + c \qquad (5.\ 4)$$

となり，推定するパラメータは半分の３個に減る．マッチングにおける各エリアの検出点数を６０画素，検索範囲を水平方向－１２～＋１１画素，垂直方向－５～＋４ラインとすれば，図５．２５よりマッチング処理が１フィールドで行える．メモリ書き込み，結果の解析を合わせ３フィールドで処理が完了する．従って，３プロセッサで１検出エリアの動ベクトル検出が行え，１２プロセッサで４エリアの動ベクトル検出ができる．このやり方では，８個の方程式から３個のパラメータを最小二乗推定することになる．映像合成処理（４フィールド・ディレイも含む），ＲＧＢからＹを作る処理，キーを変形しソフトキー化する処理に必要な３６プロセッサを合わせ，合計３クラスタを使用した．

撮影映像を用いて映像合成実験を行ったところ，検出精度はあまり良くなかった．シミュレーション実験に比べ，演算点数がかなり少ないためと思われる．検出点の位置をテクスチャがはっきりしているところなどマッチングがとり易いところ（顔の場合は目頭など）に設定すれば，精度が若干上がるが精度向上のためには，プロセッサ数を増やす必要がある．なお，評価関数

として，距離最小を示す

$$E=\Sigma\{(X_i-a\cdot x_i-b)^2+(Y_i-a\cdot y_i-c)^2\} \qquad (5.5)$$

を用いた．

（3）応用例－タイトル合成－

上記予備実験を踏まえ，具体的な応用例を試みた．ニュース番組などでは，

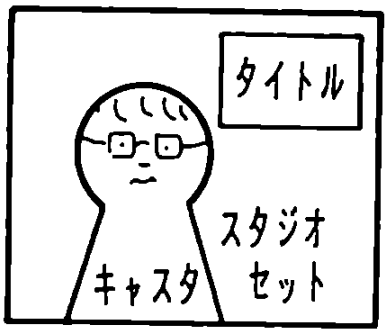


図5．28　ニュースなどでよく見られるシーン

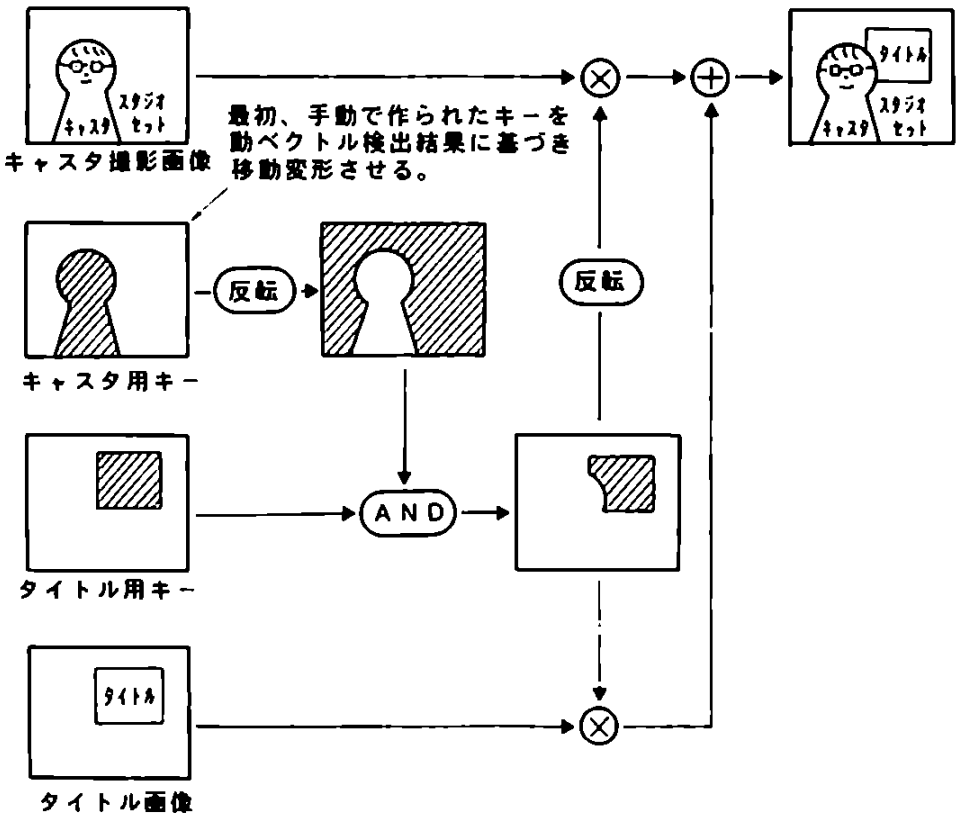
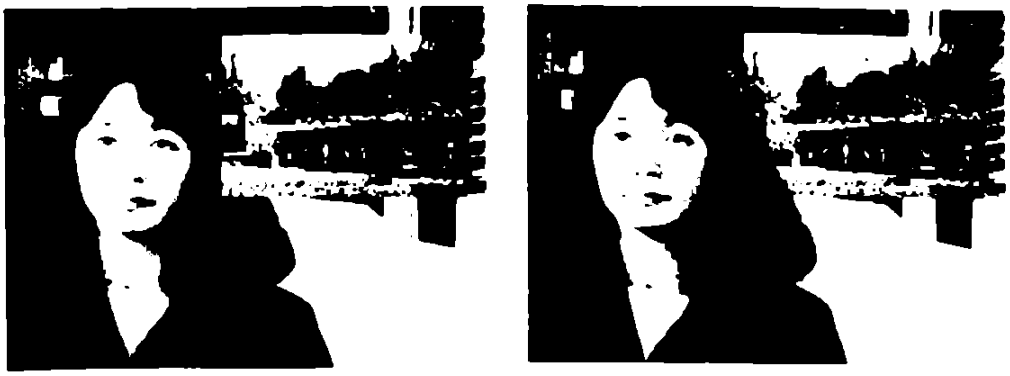


図5．29　タイトルをキャスクと背景の中間に
位置するように見せるキー合成



（ａ）通常のタイトル合成　　（ｂ）提案した方法によるタイトル合成
（頭部とタイトルの境界に注目）

図５．３０　合成画像

図５．２８に示すようなキャスタとタイトルを配したシーンがよく見られる．この場合，通常キャスタを撮った映像の上にタイトルを重ねているために，キャスクが動くとキャスタがタイトルの後方に隠れてしまう．このシーンのイメージとしては，キャスタと背景の中間にタイトルが存在するかのように見えるのが自然である．これを実現するために，図５．２９に示すようにキャスタ用のキーとタイトル用のキーを組み合わせて合成する．このうち，キャスタ用のキーを，動ベクトルを検出し移動・変形して作れば，キャスタが動いても，キャスタの後ろにタイトルがあるように見せることができる．本手法をＰｉｃｏｌ－ｓｙｓｔｅｍによりビデオレートで処理した例を図５．３０に示す．２．２で仮定した動きに必ずしも適合しない人間が抽出対象であるため，キャスタを切り出すキーが完全には得られない．この誤差を補うために，キャスタとタイトルの境界をハードキーではなくソフトキーで処理した．これらの要因によりキャスタとタイトルの境界部が曖昧であるが，見タイトルがキャスタと背景の中間に位置しているように見せることができた．

[6] 結言論

画像・映像信号をプログラムによりビデオレートで処理できる実時間映像信号処理システムについて述べた．本論文の要旨をまとめると，以下のようになる．

第１章：研究の背景と意義について述べた．

第２章：放送用の映像信号処理の特質について考察を加えた．特に，幾何学変換，効果波形発生，映像合成を取り上げ，放送用映像信号処理装置が具備すべき演算機能，演算精度について明らかにした．

第３章：プログラム可能なビデオレート映像信号処理システムのアーキテクチャについて検討した．放送用の実時間処理システム用に，マルチプロセッサ型の並列処理方式，各プロセッサが処理に必要なデータを全て確保しておく分散型のメモリ構成法，データに同期情報を付加した映像データによるプロセッサ間同期方法，画面上に乱れを生じないプログラムロード切り替え方式などについて提案した．

第４章：第２章，第３章の検討結果を踏まえて，実験システムＲＴＶＰを開発し，各種映像信号処理を適用して評価した．ＲＴＶＰは，プログラム可能なビデオレート映像信号処理システムの実現性の検証を行うために試作した実験システムのため，標本化周波数３．５８ＭＨｚの白黒映像信号２系統しか扱えないものであったが，映像転送路，分散型メモリ構成法などにより，プログラム可能なビデオレート映像信号処理システムの実現の目処が得られるとともに，実用化する上での問題点，改善すべき点などについての知見が得られた．

第５章：実験システムＲＴＶＰの試作によって得られた知見に基づき，１４．３ＭＨｚの速度で複数のカラー動画像をビデオレート処理可能なマルチプロセッサであるＰｉｃｏｔ－ｓｙｓｔｅｍを開発した．この際，小型化のためにプロセッサＬＳＩとネットワークＬＳＩの２種のＬＳＩを開発した．さらに，Ｐｉｃｏｔ－ｓｙｓｔｅｍをベースにビデオスイッチャや画質補償装置などを開発した．Ｐｉｃｏｔ－ｓｙｓｔｅｍは，動画像・映像信号処理のアルゴリズムの研究に使用されているばかりでなく，ＮＨＫスペシャル（ＮＨＫ），ソウルオリンピック（ＮＨＫ），大阪国際女子マラソン（関西ＴＶ）などの番組にも活用された．また，Ｐｉｃｏｔ－ｓｙｓｔｅｍを構成する２種のＬＳＩもそれぞれＰｉｃｏｔ－ｓｙｓｔｅｍ以外の機器に組み込まれて使用されている．

研究を開始した当初は，映像信号をプログラム可能な装置でビデオレート処理することは試みられていなかったが，現在は各所で行われるようになっ

てきている．研究の方向性に誤りがなかったことが確認された形となった．現在多くのシステムがパーソナルコンピュータのように各所に使われるようなレベルにまで達していないが，本研究は先駆けとしても意味があったと考える．

映像信号をプログラムで処理することの意義は，ソウルオリンピック放送などでの使用で確認されたが，まだ放送現場で活用されているとは言い難い．ハードウェア信仰の強い現場での意識の変革も必要であるが，欲しい機能が簡単に実現できる開発環境を整えることも必要である．特に，緊急を要する仕事が多い番組制作現場では，マンマシンインタフェースの充実が重要であろう．

現在，ＨＤＴＶレートまで処理可能なシステムの開発に着手している．今後は，映像ワークステーション，すなわち標準ＴＶ，ＨＤＴＶなどあらゆる画像・映像信号統合的にビデオレート処理でき，かつワークステーションのようにウィンドウやマウスなどを使って手軽に扱えるヒューマンインタフェースの優れたコンパクトなシステムの構築が必要であろう．

[謝辞]

本論文をまとめるに当たって，ご指導いただいた京都大学工学部応用システム科学教室英保茂教授，八村広三郎助教授に深謝いたします．

本研究はＮＨＫ放送技術研究所において行ったものである．本研究を進めるに当り当初からご指導いただいたＮＨＫ放送技術研究所画像研究部村上敬之助主任研究員，榎並和雅主任研究員に感謝します．また，Ｐｉｃｏｔ－ｓｙｓｔｅｍの開発を共同で進めていただいたＮＨＫ放送技術研究所画像研究部福井一夫主任研究員，ＴＶ方式研究部矢島亮一氏，ご協力いただいた(株)東芝小向工場の佐々木信之主務，星野浩二氏，春川和弘氏，小幡勝氏，高田龍一郎氏，金野雄二氏，船橋清一郎氏，同総合研究所嶌比呂志研究主務，同半導体事業本部山田秀喜氏をはじめ関係各位に感謝いたします．

本研究を進めるに当たり有益な御助言いただいたＮＨＫ放送技術研究所香沢淳之助主幹研究員，画像研究部二宮佑一部長，藤原正雄前副部長，野口英男副部長，同視覚情報研究部三橋哲雄主任研究員，矢野澄男研究員，同物性素子研究部小林希一主任研究員，明智和幸研究員，同技術局大場吉延前副部長，金井清晶前副部長，川野顯一郎氏，同放送技術局三上繁実氏，鈴木彰氏，武藤光明氏，指田忠男氏をはじめ関係各位に感謝したします．また，日頃御討論いただく画像研究部の皆様にも感謝いたします．

本研究の機会を与えて頂いたＮＨＫ放送技術研究所杉本昌穂前所長，泉竹博所長，吉川重夫前次長，西沢台次担当局長，吉野武彦担当局長，横山克哉前主任研究員（現，日本ビクター），石田顯一前次長（現，ＮＨＫ－ＥＳ）に深謝いたします．

[参考文献]

(1) 木戸出,坂上:"パイプライン方式と完全並列方式が増えた最近の画像処理装置",日経エレクトロニクス,1982.07.19号,p.179-212(1982)

(2) 小特集:"最近のディジタルテレビジョン技術",ＴＶ誌,Vol.36,No.1,pp.2-34(1982)

(3) 特集:"ＤＳＰ（ディジタル信号処理プロセッサ）",情報処理,Vol.30,No.1,pp.1290-1375(1989)

(4) 榎並,田中,村上:"実時間映像信号処理システムのアーキテクチャ",信学論,Vol.J68D,No.4,pp.885-892(1985)

(5) 八木,矢島,榎並,他:"実時間映像信号処理ＬＳＩ－Ｐｉｃｏｔのアーキテクチャ",信学論,Vol.J72C-II,No.5,pp.346-353(1989)

(6) 八木,矢島,榎並,他:"実時間動画像処理システムPicot-system",情処研報,CV58-2(1989)

(7) 川野,八木,福井,他:"番組制作におけるＰＩＣＯＴの実用化",1989年ＴＶ全大,14-1(1989)

(8) 八木:"ディジタル信号処理デバイスと基本機能回路",ＮＨＫ－ＥＳセミナ〔テレビジョン信号・ディジタル処理技術〕資料,p.68-101(1990)

(9) "Encoding Parameters of Digital Television for Studios",CCIR Rec.601(1982)

(10) "1125/60高精細度テレビジョン方式スタジオ規格",放送技術開発協議会,BTA S-001(1986)

(11) 日本放送協会編:"放送におけるディジタル技術",p.163-219,日本放送出版協会(1982)

(12) ＮＨＫ放送技術研究所:"ディジタルテレビ技術",p.259-277,日本放送出版協会(1990)

(13) 金次:"映像の幾何学変換用フィルタBLIF",第17回画像工学コンファレンス,pp.237-240(1986)

(14) 飯田,岡田,西村,他:"実時間３次元特殊効果装置",ＴＶ技報,IPA87-3(1987)

(15) 八木,福井,榎並,他:"映像信号処理ＬＳＩ－Ｐｉｃｏｔの演算回路に関する一検討",ＴＶ誌,Vol.44,No.9,pp.1269-1278(1990)

(16) Pratt:"Digital image processing",p.93-120,Wiley-Interscience Publication(1987)

(17) 谷内田編:"コンピュータビジョン",p.51-54,丸善(1990)

(18) 榎並.八巻.八木.他:"画像信号の幾何学変換のための補間フィルタと画質に関する一考察".信学論.Vol.J69D.No.11.pp.1617-1623(1986)
(19) 日下.湯山:"テレビジョンシステム評価用ディジタル標準画像".TV誌.Vol.39.No.10.pp.160-163(1985)
(20) 和田.大山.今井編:"感覚知覚心理学ハンドブック".p206-223.誠信書房(1969)
(21) 二宮:"画像信号処理と視覚特性".テレビ誌.Vol.40.No.4.pp.280-286(1986)
(22) 日本放送協会編:"テレビジョン番組制作技術".p.81-107.日本放送出版協会(1983)
(23) 日本放送協会編:"テレビジョン番組制作技術".p.99-102.日本放送出版協会(1983)
(24) 日本放送協会編:"テレビジョン番組制作技術".p.128-133.日本放送出版協会(1983)
(25) 吉田.桐井:"ビデオマットによる動画合成技術".TV技報.TEBS88-19(1988)
(26) 谷内田編:"コンピュータビジョン".p.167-188.丸善(1990)
(27) M.Hoetter:"Differential estimation of the global motion parameters zoom and pan".Signal Processing.Vol.16.No.3.pp.249-265(1989)
(28) 吹抜:"画像のディジタル信号処理(増補版)".p.221-227.日刊工業新聞社(1985)
(29) 二宮:"フレーム間符号化における動き補正".信学技報.IE78-6(1978)
(30) 二宮.鳶.杉山:"MUSE用動きベクトル検出装置".TV技報.TEBS103-5(1985)
(31) 情報処理学会編:"情報処理ハンドブック".p.123.オーム社(1989)
(32) 榎並.井沢.平林.鈴木:"汎用動画像シミュレータ".信学技報.IE87-73(1987)
(33) 八木.田中.榎並:"初期画像を補助手段とした動きベクトル検出法の一検討".昭63信学春季全大.D-517(1988)
(34) 田中.八木.榎並:"動きベクトルを用いた映像合成手法".1988年TV全大.19-6(1988)
(35) 田中.八木.榎並:"大きさ・向きを変える物体の動きベクトル検出法の一検討".昭64信学春季全大.D-109(1989)
(36) 西川.二宮.茨木:"最適化".p.83-93.岩波書店(1982)
(37) 八木.田中.榎並:"動物体抽出法に関する一検討".1990年TV全大.

13-8(1990)
(38) 八木.田中.櫻並:"動き抽出法に関する一検討".ＴＶ技報.ICS91-24.AIPS91-4(1991)
(39) 坂上.木戸出:"イメージプロセッサの最近の動向".信学誌.Vol.67.No.1.pp.90-98(1984)
(40) 前田:"画像処理マシン".情報処理.Vol.28.No.1.pp.19-26(1987)
(41) 吉田.長谷部:"高速画像処理技術".ＴＶ誌.Vol.42.No.3.pp.241-246(1988)
(42) 櫻並.八木:"映像信号の汎用リアルタイム処理".ＴＶ誌.Vol.43.No.12.pp.1314-1319(1989)
(43) R.Lougheed.D.McCubbrey.S.Sternberg:"CYTOCOMPUTER:Architecture for parallel image processing".Workshop Picture Data Description and Management.pp.281-286(1980)
(44) 直井.古明地.太田.他:"構造可変型ビデオレートカラー画像処理システム「韋駄天」".信学論.Vol.J73D-Ⅱ.No.10.pp.1751-1760(1990)
(45) 天満.溝口.花木:"可変パイプライン方式の画像処理プロセッサＴＩＰ－１".信学論.Vol.J68D.No.4.pp.853-860(1985)
(46) K.Luetjen.P.Gemmer.H.Ischen:"FLIP:A flexible multiprocessor system for image processing".5th ICPR.pp.326-328(1980)
(47) K.E.Batcher:"Design of a massively parallel processor".IEEE Tran.Com.Vol.C-29.No.9.pp.836-840(1980)
(48) M.J.B.Duff:"Review of the CLIP4 image processing system".AFIPS Conference.National Computer Conference.pp.1055-1060(1984)
(49) 近藤.杉山.中島:"２次元アレープロセッサ（ＡＡＰ２）とプログラミング言語".信学論.Vol.J71D.No.8.pp.1399-1406(1988)
(50) 木戸出.白男川.角谷.他:"高速画像処理装置ＴＯＳＰＩＸ－Ⅱ".第15回画像工学コンファレンス論文集.pp.133-136(1984)
(51) 福島.小林.平沢.他:"画像処理ＶＬＳＩ－Image Signal Processorのアーキテクチャ".信学論.Vol.J66C.No.12.pp.959-966(1983)
(52) 森.丸山.斎野.他:"リアルタイム画像プロセッサRISP-Ⅱ".信学技報.SSD86-96(1986)
(53) K.Inagaki.T.Kato.T.Hiroshima.et al:"MACSYM:A hierarchical parallel image processing system for event-driven pattern understanding of documents".Pattern Recognition.Vol.17.No.1.pp.85-108(1984)

(54) T.Kushner,A.Y.Wu,A.Rosenfeld:"Image processing on ZMOB", Computer Architecture for Pattern Analysis and Image Database Management,pp.88-95(1981)
(55) 長谷部,加藤,伊藤,他:"マルチプロセッサ型画像処理システムＳＩＰＳ",情処研報,CV39-5(1985)
(56) 政木,堀,内野,他:"リアルタイム画像処理システムＲＡＰＩＤ",情処研報,CV50-2(1987)
(57) I.Tamitani,H.Harasaki,T.Nishitani,et al:"A real-time video signal processor suitable for motion picture coding applications",IEEE Trans.,Vol.CAS-36,No.10,pp.1259-1266(1989)
(58) 高橋:"並列処理のためのプロセッサ結合方式",Vol.23,No.3, pp.201-209(1982)
(59) 黒川,相磯:"並列処理の諸問題 -結合方式",情報処理,Vol.27,No.9, pp.1005-1021(1986)
(60) 中川,小林,相磯:"データ駆動型離散系シュミレータＫＤＳＳ－Ｉ", 信学論,Vol.J65D,No.3,pp.386-393(1982)
(61) 小畑,金田,前川:"ブロードキャストメモリ結合形マルチマイクロプロセッサシステムの試作",情処学論,Vol24,No.3,pp.351-356(1983)
(62) 榎並,八木,村上:"２段階時間軸処理方式の提案と実時間映像処理システムＲＴＶＰへの適用",昭和60信学部門全大,171(1985)
(63) 田中,榎並,村上:"実時間映像処理システムＲＴＶＰ 制御とソフトウェア",昭和60信学全大,1276(1985)
(64) 八木,榎並,村上:"実時間映像処理システム（ＲＴＶＰ）による各種画像処理",昭和60信学部門全大,172(1985)
(65) 久野:"画像処理･画像理解におけるベンチマーク",情報処理, Vol.31,No.3,pp343-351(1990)
(66) K.Preston:"The abingdon cross benchmark survey",IEEE Computer, Vol.22,No.7,pp9-18(1989)
(67) 八木,福井,榎並,他:"動画像を実時間で処理するシステムにおける処理遅れと同期の問題についてＰｉｃｏｔ－ｓｙｓｔｅｍの対応法", 昭64信学春季全大,D-264(1989)
(68) 八木,矢島,榎並,他:"実時間映像信号処理ＬＳＩ－Ｐｉｃｏｔ", 信学技報,ICD88-35(1988)
(69) N.Yagi,R.Yajima,K.Enami,et al:"Chip set for real-time video signal processing",IAPR Workshop on Computer Vision, pp.237-240(1988)

(70) 斉藤.上田.佐々木:"プログラム動作を行うプロセッサによる映像信号処理（ワイプキー発生器への適用）",放送技術.Vol.44.No.3.pp.217-222(1991)
(71) 特集:"放送用大規模映像・音声分配システム",東芝レビュー.Vol.45.No.12.pp.927-956(1990)
(72) 高橋.萬:"ＮＨＫ放送センタディジタル切替.分配システム",ＴＶ技報.BF091-2(1991)
(73) 磯村.松尾.広田:"ＪＳＢのディジタル分配システム",ＴＶ技報.BF091-13(1991)
(74) M.Ott.榎並.羽鳥.他:"ＰｉｃＰＥｎ－実時間映像信号処理装置Ｐｉｃｏｔのためのプログラミング環境",ＴＶ誌.Vol.44.No.11.pp.1570-1578(1990)
(75) M.Ott.N.Yagi.K.Fukui.et al:"Timing in synchronous system",1989年ＴＶ全大.14-4(1989)
(76) N.Yagi.K.Fukui.K.Enami.et al:"Real-time video signal processing system for dynamic images",SPIE Vol.1199.Proc.Visual Communications and image processing IV.pp.866-877(1989)
(77) 八木.福井.榎並.他:"実時間動画像処理システムＰｉｃｏｔの処理能力評価",1989信学秋季全大.D-107(1989)
(78) 三上.榎並.八木.他:"Ｐｉｃｏｔシステムを用いたディジタルビデオスイッチャの開発",1988年ＴＶ全大.19-2(1988)
(79) 佐々木.小暮.富田.他:"ＰＩＣＯＴを用いた新しいスイッチャ効果の開発",1989年ＴＶ全大.14-2(1989)
(80) 八木.福井.榎並.他:"実時間動画像処理システムＰｉｃｏｔとその応用",画像電子学会研究会予稿.89-01-06(1989)
(81) 指田.武藤.川野:"Ｐｉｃｏｔの現場での運用",ＴＶ技報.BF089-6(1989)
(82) 林.八木.福井.他:"Ｐｉｃｏｔによるノイズ軽減処理",1989年ＴＶ全大.14-3(1989)
(83) 林.井上.藤原.他:"時間軸圧縮による映像２チャンネル同時伝送装置",ＴＶ誌.Vol.35.No.8.pp.631-636(1981)
(84) 末岡.大場.平川:"２画面伝送装置",ＴＶ誌.Vol.43.No.2.pp.132-134(1989)
(85) 八木.福井.榎並.他:"ＣＧキャラクタの撮影画像を用いた実時間操作",1990年信学秋季全大.D-416(1990)
(86) 佐々木.富田.並川.他:"Ｐｉｃｏｔを用いたマラソンランナーのピッチ.

ストライド計測システムの開発", 1990年ＴＶ全大, 12-3 (1990)
1990年ＴＶ全大, 12-4 (1990)

[発表論文等リスト]

[学会論文]

・榎並.田中.村上:"実時間映像信号処理システムのアーキテクチャ".
信学論.Vol.J68D.No.4.pp.885-892(1985)
(上記文献中の著者名"田中"は、八木の旧姓)

・榎並.八巻.八木.他:"画像信号の幾何学変換のための補間フィルタと画質に関する一考察".信学論.Vol.J69D.No.11.pp.1617-1623(1986)

・K.Murakami.K.Enami.N.Yagi:"A proposed universal signal-processing system".SMPTE Journal.Vol.96.No.6.pp.527-531(1987)

・K.Enami.N.Yagi.K.Murakami:"Real-time video signal processor".
SMPTE Journal.Vol.96.No.12.pp.1158-1165(1987)

・榎並.八木.矢島.他:"汎用映像信号処理システムＰｉｃｏｔシステム".
ＴＶ誌.Vol.43.No.2.pp.178-186(1989)

・八木.矢島.榎並.他:"実時間映像信号処理ＬＳＩ－Ｐｉｃｏｔの
アーキテクチャ".信学論.Vol.J72C-Ⅱ.No.5.pp.346-353(1989)

・N.Yagi.R.Yajima.K.Enami.et al:"An architecture of real-time video signal processing LSI-Picot".Electronics and Communications in Japan.Part2.Vol.73.No.6.pp.70-78(1990)

・八木.福井.榎並.他:"映像信号処理ＬＳＩ－Ｐｉｃｏｔの演算回路に関する一検討".ＴＶ誌.Vol.44.No.9.pp.1269-1278(1990)

・M.Ott.‥.八木.他:"ＰｉｃＰＥｎ－実時間映像信号処理装置Ｐｉｃｏｔのためのプログラミング環境".ＴＶ誌.Vol.44.No.11.pp.1570-1578(1990)

・N.Yagi.K.Fukui.K.Enami.et al:"Programmable real-time video signal processing system".SMPTE Journal.Vol.100.No.10(1991)

・八木.田中.榎並:"映像合成のための動きベクトル検出法に関する一検討".
ＴＶ誌.Vol.45.No.10(1991)

[国際会議]

・K.Enami.N.Yagi.K.Murakami:"Architecture of a real-time video signal processor".ICASSP'86.pp.793-796(1986)

・K.Enami.N.Yagi.K.Murakami:"Universal video signal processing system".128th SMPTE technical conference(1986)

・N.Yagi.R.Yajima.K.Enami.et al:"Chip set for real-time video signal processing".IAPR Workshop on Computer Vision.pp.237-240(1988)

・N.Yagi.K.Fukui.K.Enami.et al:"Real-time video signal processing system for dynamic images".SPIE Vol.1199.Proc.

Visual Communications and image processing IV, pp.866-877(1989)
・K.Fukui, K.Enami, N.Yagi, et al:"Picot:picture computer",
132nd SMPTE technical conference(1990)
[解説]
・榎並, 八木:"映像信号の汎用リアルタイム処理",
ＴＶ誌, Vol.43, No.12, pp.1314-1319(1989)
・八木, 榎並:"映像機器構成法の新しい流れ",
放送技術, Vol.44, No.4, pp.374-381(1991)
・榎並, 八木:"並列リアルタイム映像信号処理システム",
信学誌, Vol.74, No.6, pp.573-576(1991)
・八木, 福井, 榎並, 他:"映像コンピュータＰｉｃｏｔ", ＮＨＫ技研Ｒ＆Ｄ,
No.16(1991)
[学会研究会]
・榎並, 八巻, 八木, 他:"画像補間フィルタと画質の評価",
画像電子学会研究会予稿, 86-01-01(1986)
・八木, 矢島, 榎並, 他:"実時間映像信号処理ＬＳＩ－Ｐｉｃｏｔ",
信学技報, ICD88-35(1988)
・八木, 矢島, 榎並, 他:"実時間動画像処理システム
Ｐｉｃｏｔ-ｓｙｓｔｅｍ", 情処研報, CV58-2(1989)
・八木, 福井, 榎並, 他:"実時間動画像処理システムＰｉｃｏｔとその応用",
画像電子学会研究会予稿, 89-01-06(1989)
・佐々木, 小暮, 八木, 他:"汎用映像処理システムＰｉｃｏｔの開発",
ＴＶ技報, BFO89-5(1989)
・八木, 田中, 榎並:"動き抽出法に関する一検討",
ＴＶ技報, ICS91-24, AIPS91-4(1991)
・榎並, 福井, 八木, 他:"実時間映像信号処理装置Ｐｉｃｏｔ", ＴＶ技報,
IPCV91-13, AIPS91-24(1991)
[学会全国大会]
・榎並, 田中, 村上:"実時間映像処理システムＲＴＶＰ
ハードウェア構成", 昭和60信学全大, 1277(1985)
・田中, 榎並, 村上:"実時間映像処理システムＲＴＶＰ
制御とソフトウェア", 昭和60信学全大, 1276(1985)
・村上, 榎並, 田中, 他:"放送における映像処理汎用システムの提案と試作",
昭60ＴＶ全大, 8-5(1985)
（上記３文献中の著者名"田中"は, 八木の旧姓）
・八木, 榎並, 村上:"実時間映像処理システム（ＲＴＶＰ）による

各種画像処理", 昭和60信学部門全大, 172(1985)
・榎並, 八木, 村上:"２段階時間軸処理方式の提案と実時間映像処理システムＲＴＶＰへの適用", 昭和60信学部門全大, 171(1985)
・榎並, 八巻, 八木, 他:"画像補完法と画質に関する一考察", 昭61信学全大, 1240(1986)
・榎並, 八木, 矢島, 他:"実時間映像信号処理システムの提案Ｐｉｃｏｔシステム", 昭63信学全大, D-201(1988)
・八木, 矢島, 榎並, 他:"実時間映像信号処理ＬＳＩ－Ｐｉｃｏｔ", 昭63信学全大, D-202(1988)
・八木, 矢島, 榎並, 他:"実時間映像信号処理ＬＳＩ－制御機構", 昭63信学全大, D-203(1988)
・榎並, 八木, 矢島, 他:"実時間映像信号処理用並列プロセッサ結合ＬＳＩ", 昭63信学全大, D-204(1988)
・矢島, 八木, 榎並, 他:"実時間映像信号処理システムＰｉｃｏｔにおける処理方式", 昭63信学全大, D-205(1988)
・八木, 田中, 榎並:"初期画像を補助手段とした動きベクトル検出法の一検討", 昭63信学春季全大, D-517(1988)
・八木, 矢島, 榎並, 他:"実時間映像信号処理装置Ｐｉｃｏｔシステムの放送への応用", 1988年ＴＶ全大, 19-1(1988)
・三上, 榎並, 八木, 他:"Ｐｉｃｏｔシステムを用いたディジタルビデオスイッチャの開発", 1988年ＴＶ全大, 19-2(1988)
・榎並, 八木, 矢島, 他:"Ｐｉｃｏｔシステムを用いたディジタルビデオスイッチャの制御", 1988年ＴＶ全大, 19-3(1988)
・矢島, 八木, 榎並, 他:"Ｐｉｃｏｔシステムにおけるソフトウェア開発", 1988年ＴＶ全大, 19-4(1988)
・田中, 八木, 榎並:"動きベクトルを用いた映像合成手法", 1988年ＴＶ全大, 19-6(1988)
・八木, 福井, 榎並, 他:"動画像を実時間で処理するシステムにおける処理遅れと同期の問題についてＰｉｃｏｔ－ｓｙｓｔｅｍの対応法", 昭64信学春季全大, D-264(1989)
・田中, 八木, 榎並:"大きさ･向きを変える物体の動きベクトル検出法の一検討", 昭64信学春季全大, D-109(1989)
・川野, 八木, 福井, 他:"番組制作におけるＰＩＣＯＴの実用化", 1989年ＴＶ全大, 14-1(1989)
・佐々木, ･･･, 八木, 他:"ＰＩＣＯＴを用いた新しいスイッチャ効果の開発", 1989年ＴＶ全大, 14-2(1989)

・林.八木.福井.他:"Ｐｉｃｏｔによるノイズ軽減処理".
1989年ＴＶ全大.14-3(1989)
・M.Ott.N.Yagi.K.Fukui.et al:"Timing in synchronous system".
1989年ＴＶ全大.14-4(1989)
・八木.福井.櫻並.他:"実時間動画像処理システムＰｉｃｏｔの
処理能力評価".1989信学秋季全大.D-107(1989)
・福井.八木.櫻並:"Ｐｉｃｏｔによるステレオ画像処理とその--応用例".
1990年ＴＶ全大.12-5(1990)
・佐々木.八木.福井.他:"Ｐｉｃｏｔを用いた並列処理の各種例と分類".
1990年ＴＶ全大.12-6(1990)
・八木.田中.櫻並:"動物体抽出法に関する--検討".
1990年ＴＶ全大.13-8(1990)
・八木.福井.櫻並.他:"ＣＧキャラクタの撮影画像を用いた実時間操作".
1990年信学秋季全大.D-416(1990)
・八木.櫻並:"動ぼけを考慮した映像合成法".
1991年信学春季全大.D-454(1991)
・猪山.福井.八木.他:"両眼立体視によるリアルタイム映像抽出手法".
1991年信学春季全大.D-493(1991)

[付録 1] プロセッサ用高級言語仕様概説

1. プログラムの形式

・プログラムの形式は以下の通り.

```
(例) picot1{          'プロセッサの指定
     'MODULE1':       'モジュール名の指定 (''の中,24char以内)
         dlc(0):      '以下, 実行文
         ldr(hint1,0):
         ldr(hinc1,1,0):
         mgs(greg1,hcmp2):
         while(svd) out(clr):
       loop:
         while(shd) au(hint1/reg1,hinc1,hm0,hoff):
         au(hau/reg1,hinc1,hm0,hoff):
         while(!shd) {
           while(!hauf2) out(1a):
           while( hauf2) out(1b):
         }
         if(!svd) goto loop:
         return:
     }                'プロセッサ1のプログラムMODULE1の終り
     picot2{          'プロセッサの指定
     'MODULE2':       'モジュール名の指定 (''の中,24char以内)
         ..........   '以下, 実行文
         ..........
         ..........
     }                'プロセッサ2のプログラムの終り
```

このように, ひとつのソースファイル中に複数のプログラムが記述できる.
例えば, picot1が複数存在しても, かまわない.

・コメント, モジュール名以外は, 小文字を使用する.

・各行の ' 以降は, コメントが記述できる.

2. 制御命令

・制御命令には以下のものがある.

goto LABEL

LABELのついた文へ無条件にジャンプする.
ラベルは, label:のように記述する.

```
(例) loop:
          ..........
          goto loop;  'loopにジャンプする.
```

if (FLAG) goto LABEL

FLAGがonなら, LABELのついた文にジャンプする.

FLAG

shd: システムのHD, svd: システムのVD, sfd: システムのFP
ahd: 入力1AのHD, avd: 入力1AのVD, afd: 入力1AのFP
bhd: 入力1BのHD, bvd: 入力1BのVD, bfd: 入力1BのFP
z: ALUのゼロ・フラッグ, pz: ALUのプラス／ゼロ・フラッグ
hauf1: HAUのフラッグ1, hauf2: HAUのフラッグ2

```
      vaufl : VAUのフラッグ１，vauf2 : VAUのフラッグ２
    上記の条件FLAGの否定として，!が使える．（例：!shd.!svdなど）
  （例）loop:
            ..........
            if (shd) goto loop;  'shdのときloopにジャンプする．
while (FLAG) FUNC_STATEMENT
  FLAGがonの間，FUNC_STATEMENTが実行される．
    FLAG
      shd : システムのHD，svd : システムのVD，sfd : システムのFP
      ahd : 入力IAのHD，  avd : 入力IAのVD，  afd : 入力IAのFP
      bhd : 入力IBのHD，  bvd : 入力IBのVD，  bfd : 入力IBのFP
      z : ALUのゼロ・フラッグ，pz : ALUのプラス／ゼロ・フラッグ
      haufl : HAUのフラッグ１，hauf2 : HAUのフラッグ２
      vaufl : VAUのフラッグ１，vauf2 : VAUのフラッグ２
    上記の条件FLAGの否定として，!が使える．（例：!shd.!svdなど）
  （例）while(shd) alu(･･･);  'shdの間 alu(･･･) を実行する．
while (FLAG) {FUNC_STATEMENTS}
  FLAGがonの間，FUNC_STATEMENTSが実行される．
  while文は，２重の入れ子まで許される．
    FLAG
      shd : システムのHD，svd : システムのVD，sfd : システムのFP
      ahd : 入力IAのHD，  avd : 入力IAのVD，  afd : 入力IAのFP
      bhd : 入力IBのHD，  bvd : 入力IBのVD，  bfd : 入力IBのFP
      z : ALUのゼロ・フラッグ，pz : ALUのプラス／ゼロ・フラッグ
      haufl : HAUのフラッグ１，hauf2 : HAUのフラッグ２
      vaufl : VAUのフラッグ１，vauf2 : VAUのフラッグ２
    上記の条件FLAGの否定として，!が使える．（例：!shd.!svdなど）
  （例）while(svd){  'svdの間 { から } を実行
          .....
          alu(･･･);
          .....
        }
        while(svd){
          .....
          while(shd) alu(･･･);  'shdかつsvdの間alu(･･･)を実行
          .....
        }
        while(svd){
          .....
          while(shd){  'shdかつsvdの間 { から } を実行する．
            .....
          }
          .....
        }
return
  ルーチンから抜け出る．インストラクションセットのRESTARTを実行する．
  （例）return;  'RESTART命令が実行される．
```

3. ファンクション命令

nop

何もしない

out(MPX)

出力を制御する関数

MPX

la	: 外部入力IA	lb	: 外部入力IB
alu	: ALUの出力	aluf	: ALUのフラッグ
mpy	: 乗算器の出力	dm	: Date Memoryの出力
hau	: HAUの出力	vau	: VAUの出力
greg	: 汎用レジスタ	clr	: 0
h	: ＨＤ	V	: ＶＤ
VH	: ＶＤとＨＤ	FV	: ＦＰとＶＤ
FVH	: ＦＰとＶＤとＨＤ		

（例）out(alu)： ・出力OAにALUの出力を出力する.

alu(INA,INB,FUNC,FL,L/H)

算術論理演算器を制御する関数

INA　入力選択

la	: 外部入力IA	lb	: 外部入力IB
mpy	: 乗算器の出力	hauh	: HAUの出力
acons	: レジスタ	dm	: Date Memoryの出力

INB　入力選択

la	: 外部入力IA	alu	: ALUの出力
mpy	: 乗算器の出力	vauh	: VAUの出力
dm	: Date Memoryの出力		

FUNC　演算機能

clr	: クリア (all 0)	prt	: プリセット (all 1)
sba	: b-a	sab	: a-b
add	: a+b	xor	: a^b
or	: a|b	and	: a&b
max	: max(a,b)	min	: min(a,b)
absa	: |a|	absb	: |b|
pa	: a	pb	: b
ma	: -a	mb	: -b

FL　フラッグ

z	: zeroフラッグ	pz	: plus/zeroフラッグ

L/H　演算結果のラッチ，ホールド

lch	: ラッチ	hld	: ホールド

（例）alu(lb,la,add,z,lch)： ・LB＋LAを計算し，ラッチする.
・zeroフラッグを選択.

mpy(INA,INB,FMT,L/H)

乗算器を制御する関数

INA　入力選択

la	: 外部入力IA	lb	: 外部入力IB
alu	: ALUの出力	dm	: Date Memoryの出力

INB　入力選択

la	: 外部入力IA	lb	: 外部入力IB
mcons	: レジスタ	dm	: Date Memoryの出力

FMT　小数点位置
lsd: 演算モード１　　msl: 演算モード２
L/H　演算結果のラッチ，ホールド
lch: ラッチ　　hld: ホールド
（例）mpy(la,lb,lsd,lch): 'LA×LBをモード１で計算し，ラッチする．

au(HFUNC,HINC,HMODE,HAREP,VFUNC,VINC,VMODE,VAREP)

算術演算器HAUとVAUを制御する関数

HFUNC　演算機能
hnop　　: 何もしない
hreg1/reg2: hreg1→hreg2
hint1/reg1: hint1→hreg1
hint2/reg1: hint2→hreg1
hau/reg1　: hreg1 + HINC → hreg1
hau/reg2　: hreg1 + HINC → hreg2
hau/reg12 : hreg1 + HINC → hreg1,hreg2
HINC　増分値レジスタ
hinc1: HINC1　　hinc2: HINC2
HMODE　リプレイス・モード
hm1: モード１　hm2: モード２　hm3: モード３　hm4: モード４
HAREP　自動リプレイス
hoff: マニュアル・モード　hon: オート・モード
VFUNC　演算機能
vnop　　: 何もしない
vreg1/reg2: vreg1→vreg2
vint1/reg1: vint1→vreg1
vint2/reg1: vint2→vreg1
vau/reg1　: vreg1 + VINC → vreg1
vau/reg2　: vreg1 + VINC → vreg2
vau/reg12 : vreg1 + VINC → vreg1,vreg2
VINC　増分値レジスタ
vinc1: VINC1　　vinc2: VINC2
VMODE　リプレイス・モード
vm1: モード１　vm2: モード２　vm3: モード３　vm4: モード４
VAREP　自動リプレイス
voff: マニュアル・モード　von: オート・モード
（例）au(hau/reg1,hinc1,hm0,hoff,vint1/reg1,vinc1,vm1,von):
'HREG1=HREG1+HINC1，マニュアルモード０を選択．
'VREG1=VINT1，自動リプレイスモード１を選択

dm(ADR,DAT,R/W)

データメモリのアクセスを制御する関数

ADR
la　: 外部入力IA　　lb　: 外部入力IB
alu : ALUの出力　　mpy : 乗算器の出力
hau : HAUの出力　　vau : VAUの出力
tmc : ラインディレイ　　hvau : 補間パラメータ
hv9 : VAU(9bit),HAU(9bit)　　lab9 : IB(9bit),IA(9bit)
hv10: VAU(10bit),HAU(10bit)　lab10: IB(10bit),IA(10bit)
hv11: VAU(11bit),HAU(11bit)　lab11: IB(11bit),IA(11bit)

hv12 : VAU(12bit).HAU(12bit)　lab12 : IB(12bit).IA(12bit)

DAT

la : 外部入力IA　　lb : 外部入力IB

alu : ALUの出力　　mpy : 乗算器の出力

R/W

r : read　　w : write

（例）dm(hau,alu,w) ; 'アドレスHAUにALUのデータを書き込む.

dmb(BANK)

データメモリのバンクを制御する関数

BANK　バンク番号　0～255

（例）dmb(255) ; 'データメモリのバンク２５５を選択する.

dlc(DL)

遅延回路を制御する関数

DL　遅延量　0～7

（例）dlc(4) ; 'ディレイ量を４に設定する.

tmc(DL)

ラインディレイを制御する関数

DL　遅延量　7～912

（例）tmc(500) ; 'ディレイ量を５００に設定する.

ldr(SREG,DATA)

定数レジスタの値をセットする命令

SREG　定数レジスタ

hint1 : HAUの初期値レジスタ１　hint2 : HAUの初期値レジスタ２

hinc1 : HAUの増分値レジスタ１　hint2 : HAUの増分値レジスタ２

hcmp1 : HAUの比較値レジスタ１　hcmp2 : HAUの比較値レジスタ２

hrep1 : HAUの置換値レジスタ１　hrep2 : HAUの置換値レジスタ２

vint1 : VAUの初期値レジスタ１　vint2 : VAUの初期値レジスタ２

vinc1 : VAUの増分値レジスタ１　vint2 : VAUの増分値レジスタ２

vcmp1 : VAUの比較値レジスタ１　vcmp2 : VAUの比較値レジスタ２

vrep1 : VAUの置換値レジスタ１　vrep2 : VAUの置換値レジスタ２

acons : ALUのレジスタ　　mcons : 乗算器のレジスタ

（例）ldr(hint1,0) ; 'hint1に０を置数する.

（注）この命令は，強制的にcontinueになるため，while文などの制御命令中に記述することはできない.

mgs(GREG,SREG)

汎用インタフェースレジスタから定数レジスタにデータをコピーする命令

GREG　汎用インタフェースレジスタ

greg1 ～ greg7

SREG　定数レジスタ

hint1 : HAUの初期値レジスタ１　hint2 : HAUの初期値レジスタ２

hinc1 : HAUの増分値レジスタ１　hint2 : HAUの増分値レジスタ２

hcmp1 : HAUの比較値レジスタ１　hcmp2 : HAUの比較値レジスタ２

hrep1 : HAUの置換値レジスタ１　hrep2 : HAUの置換値レジスタ２

vint1 : VAUの初期値レジスタ１　vint2 : VAUの初期値レジスタ２

vinc1 : VAUの増分値レジスタ１　vint2 : VAUの増分値レジスタ２

vcmp1 : VAUの比較値レジスタ１　vcmp2 : VAUの比較値レジスタ２

vrep1 : VAUの置換値レジスタ１　vrep2 : VAUの置換値レジスタ２

acons: ALUのレジスタ　　mcons: 乗算器のレジスタ

（例）mgs(greg1.hintl): 'greg1をhintlにコピーする.

■fg(FREG,GREG)

関数レジスタから汎用インタフェースレジスタにデータをコピーする命令

FREG 関数レジスタ

alu : ALUの出力	mpy : 乗算器の出力
ia : 外部入力IA	dm : Date Memoryの出力
hau : HAUの出力	vau : VAUの出力
hreg1: HAUのREG1	vreg1: VAUのREG1
hreg2: HAUのREG2	vreg2: VAUのREG2

GREG 汎用インタフェースレジスタ

greg1 ～ greg7

（例）mfg(alu.greg1): 'aluレジスタをgreg1にコピーする.

■gg(GREG,GREG)

汎用インタフェースレジスタから汎用インタフェースレジスタにデータをコピーする命令

GREG 汎用インタフェースレジスタ

greg1 ～ greg7

（例）mgg(greg1.greg2): 'greg1をgreg2にコピーする.

〔付録２〕ネットワーク用言語仕様概説

1．プログラムの形式

・プログラムの形式は以下の通り．

```
（例）network[              'ネットワークの指定
      'MODULE1':            'モジュール名の指定（''の中，24char以内）
          net(in1,pula):    '以下，実行文
          net(in2,pulb):
          net(pu1,pu2a):
          net(pu2,pu3a):
          while(!shd) {
            net(pu2,out2):
            net(pu3,out2):
          }
          return;
      ]                     'ネットワークのプログラムMODULE1の終り
      network[              'ネットワークの指定
      'MODULE_NAME':        'モジュール名の指定（''の中，24char以内）
          .........         '以下，実行文
          .........
          .........
      ]                     'ネットワークのプログラムの終り
```

このように，ひとつのソースファイル中に複数のプログラムが記述できる．

・コメント，モジュール名以外は，小文字を使用する．

・各行の ' 以降は，コメントが記述できる．

2．制御命令

・制御命令には以下のものがある．

goto LABEL

LABELのついた文へ無条件にジャンプする．

ラベルは，label:のように記述する．

```
（例）loop:
          .........
          goto loop;  'loopにジャンプする．
```

if (FLAG) goto LABEL

FLAGがonなら，LABELのついた文にジャンプする．

FLAG　フラッグ

shd：システムのHD，svd：システムのVD，sfd：システムのFP

上記の条件FLAGの否定として，!が使える．（例：!shd,!svdなど）

```
（例）loop:
          .........
          if (shd) goto loop;  'shdのときloopにジャンプする．
```

while (FLAG) FUNC_STATEMENT

FLAGがonの間，FUNC_STATEMENTが実行される．

FLAG　フラッグ

shd：システムのHD，svd：システムのVD，sfd：システムのFP

上記の条件FLAGの否定として，!が使える．（例：!shd,!svdなど）

```
（例）while(shd) net(･･･);  'shdの間 net(･･･) を実行する．
```

while (FLAG) {FUNC_STATEMENTS}
FLAGがonの間, FUNC_STATEMENTSが実行される.
while文は, 1重の入れ子までしか許されない.
FLAG フラッグ
shd: システムのHD, svd: システムのVD, sfd: システムのFP
上記の条件FLAGの否定として, !が使える. (例: !shd,!svdなど)
(例) while(svd){ 'svdの間 { から } の命令を実行する.
.....
}
halt
その行で停止する.
(例) halt: 'ここ以下の行を実行しない.
return
ルーチンから抜け出る. インストラクションセットのRESTARTを実行する.
(例) return: 'RESTART命令が実行される.

3. ファンクション命令
・ファンクション命令には, 以下のものがある.
net(IN,OUT)
ネットワークの入出力を接続する関数
IN 入力
in1～in16 クラスタ外部からの入力
pu1～pu16 プロセッサ1～16の出力
OUT 出力
out1～out16 クラスタ外部への出力
pu1a～pu16a プロセッサ1～16の入力1A
pu1b～pu16b プロセッサ1～16の入力1B
(例) net(in1,pu1a): 'in1をpu1aにつなぐ.